Journal of Statistical Software

January 2022, Volume 101, Issue 1. *doi: 10.18637/jss.v101.i01*

The poolr Package for Combining Independent and Dependent p Values

Ozan Cinar
Maastricht University

Wolfgang Viechtbauer
Maastricht University

Abstract

The **poolr** package provides an implementation of a variety of methods for pooling (i.e., combining) p values, including Fisher's method, Stouffer's method, the inverse chi-square method, the binomial test, the Bonferroni method, and Tippett's method. More importantly, the methods can be adjusted to account for dependence among the tests from which the p values have been derived assuming multivariate normality among the test statistics. All methods can be adjusted based on an estimate of the effective number of tests or by using an empirically-derived null distribution based on pseudo replicates that mimics a proper permutation test. For the Fisher, Stouffer, and inverse chi-square methods, the test statistics can also be directly generalized to account for dependence, leading to Brown's method, Strube's method, and the generalized inverse chi-square method. In this paper, we describe the various methods, discuss their implementation in the package, illustrate their use based on several examples, and compare the **poolr** package with several other packages that can be used to combine p values.

Keywords: combining p values, dependent p values, R.

1. Introduction

> "When a number of quite independent tests of significance have been made, it sometimes happens that although few or none can be claimed individually as significant, yet the aggregate gives an impression that the probabilities are on the whole lower than would often have been obtained by chance. It is sometimes desired, taking account only of these probabilities, and not of the detailed composition of the data from which they are derived, which may be of very different kinds, to obtain a single test of the significance of the aggregate, based on the product of the probabilities individually observed." Fisher (1932, p. 99)

The quote above, taken from the 4th edition of Fisher's *Statistical Methods for Research Workers*, reveals a longstanding interest by statisticians and researchers in methods for combining the results of multiple significance tests. The goal is to combine the p values of several significance tests – which are thematically related to each other in some way – into a single p value that tests the significance of the collection as a whole. Numerous methods for this purpose have been described in the literature (for some reviews, see Becker 1994; Cousins 2008; Rosenthal 1978).

Some well-known methods include those described by Fisher (1932), Stouffer, Suchman, DeVinney, Star, and Williams Jr. (1949), and Wilkinson (1951). However, all of these methods assume that the p values to be combined are derived from independent tests. This assumption is known to be violated in many situations. For example, in genome-wide association studies (GWAS), a million or more single-nucleotide polymorphisms (SNPs) are nowadays tested for their association with some phenotype of interest. The p values from SNPs belonging to the same gene or biological pathway can be combined to shift the focus of the study to higher biological structures (Lehne, Lewis, and Schlitt 2011). However, SNPs show non-random associations known as linkage disequilibrium (LD) (Slatkin 2008), leading to dependence among the tests and hence p values. Ignoring this dependence when combining the p values could lead to overly conservative or to overly liberal Type I error rates (Alves and Hu 2014).

Methods for combining p values can be modified to account for dependence among the tests. For example, Brown (1975) proposed using a Satterthwaite approximation for the distribution of the test statistic of Fisher's method that takes the degree of dependence into consideration. Stouffer's method can also be easily generalized to dependent tests (Strube 1985). Furthermore, several authors (Cheverud 2001; Nyholt 2004; Li and Ji 2005; Gao, Starmer, and Martin 2008; Galwey 2009) have described methods to estimate the effective number of independent tests using principal component analysis (PCA) of a correlation matrix that reflects the dependence among the tests. Methods that assume independence can then be modified based on such an estimate to take the degree of dependence into consideration. Finally, it is also possible to combine dependent tests using empirically-derived null distributions using resampling procedures (Lin 2005; Liu *et al.* 2010). Although the latter is often considered a gold-standard technique, especially in the field of genomics, the increased computational burden is an important drawback (Moskvina, O'Dushlaine, Purcell, Craddock, Holmans, and O'Donovan 2011).

At the moment, there are various options available in R (R Core Team 2021) for combining p values of independent and dependent tests. A large number of methods for combining independent p values are implemented in the **metap** package (Dewey 2021). Three of these methods (Fisher's, Stouffer's, and the logit-transformation method by George 1977) are also available via the `combine.test()` function of the **survcomp** package (Schroeder, Culhane, Quackenbush, and Haibe-Kains 2011), while the **aggregation** package (Yi and Pachter 2018) also provides Fisher's method in addition to the methods by Tippett (1931) and Lancaster (1961). Independent p values can also be combined with the Fisher and Stouffer methods using the `metap()` function of the **gap** package (Zhao 2007).

For dependent tests, **EmpiricalBrownsMethod** (Poole 2021) and **CombinePValue** (Dai, Leeder, and Cui 2014) provide generalizations of Fisher's method analogous to Brown (1975), but avoid the numerical integration required by Brown's method in various ways. However, both implementations require access to the raw data, which limits their applicability to scenarios where only the p values and some kind of correlation matrix reflective of the dependence structure is available. The **TFisher** package also provides an implementation of Brown's method, and can work directly with a set of p values and a corresponding correlation matrix, but the accompanying manuscript (Zhang, Tong, Landers, and Wu 2020) does not provide any technical details about this aspect of the package. Finally, a method for combining (possibly dependent) p values was recently described by Wilson (2019), which is implemented in the **harmonicmeanp** package. We will examine all of these alternative packages in more detail further below.

The primary aim of this paper, however, is to present a new package called **poolr**, which provides a variety of methods for combining independent, and more importantly, dependent p values. The package contains six "base methods" which can be adjusted based on four different PCA-based approaches that estimate the effective number of tests. Furthermore, the package can generate empirical null distributions for the base methods to calculate a combined p value that accounts for the dependence among the tests. Finally, the package includes implementations of Brown's method, Strube's generalization of the Stouffer method, and the generalized inverse chi-square method.

The paper is structured as follows. In Section 2, we describe the methods that are implemented in **poolr**. The functions and their arguments are then explained in Section 3. An illustrative application of the **poolr** package is provided in Section 4, followed by a comparison, in Section 5, of the methods implemented in **poolr** with those in packages that also deal with the combination of p values from independent and dependent tests. Finally, Section 6 concludes the paper with a discussion of some miscellaneous topics.

2. Methods

The problem to be addressed can be succinctly stated as follows. Let p_i denote the one- or two-sided p value corresponding to the i-th null hypothesis, H_{0i}, where $i = 1, \ldots, k$. We assume that the tests being used to generate the p_i values have nominal properties (i.e., the Type I error rates are equal to the chosen α value). Therefore, we assume that $p_i \sim \text{Uniform}(0,1)$ when H_{0i} is true. The goal is to combine the p values into a single p value to test the joint null hypothesis that H_{0i} is true for all k tests,

$$H_0\colon \cap_{i=1}^{k} H_{0i},$$

versus the alternative

$$H_1 : \text{at least one } H_{0i} \text{ is false.}$$

We will denote the combined p value with p_c and use α_c to denote the desired Type I error rate to which p_c is compared to determine whether the joint null should be rejected or not.

2.1. Base methods

Six "base methods" are available in **poolr** for this task. All but one of these methods assume that the p values to be combined are independent of each other. These base methods can be adjusted to consider the degree of dependence among the tests. We will start by introducing the base methods and then continue with describing the possible adjustments.

Fisher's method

Fisher's method (Fisher 1932) is one of the most commonly used techniques for combining a set of p values. Assuming that $p_i \sim \text{Uniform}(0,1)$, it can be shown that $-2\ln(p_i)$ is chi-square distributed with 2 degrees of freedom (df). A combined test statistic can then be calculated with

$$X^2 = -2\sum_{i=1}^{k} \ln(p_i). \tag{1}$$

Since the sum of independent chi-square distributed random variables is also chi-square distributed (with degrees of freedom equal to the sum of the degrees of freedom of the summands), X^2 follows a chi-square distribution with $\text{df} = 2k$ under the joint null hypothesis. Therefore, the combined p value is given by $p_c = 1 - F(X^2, 2k)$ where $F(\cdot, f)$ denotes the cumulative distribution function of a chi-square distribution with $\text{df} = f$.

Stouffer's method

Stouffer's method (Stouffer *et al.* 1949) is another well-known method for pooling p values. Let $\Phi(\cdot)$ denote the cumulative distribution function of a standard normal distribution and $\Phi^{-1}(\cdot)$ its inverse. Given that the p values are uniformly distributed under the null hypothesis, $z_i = \Phi^{-1}(1 - p_i)$ follows a standard normal distribution, and thus

$$z = \sum_{i=1}^{k} z_i/\sqrt{k} \tag{2}$$

is also standard normal under the joint null. Note that we use $1 - p_i$ in the computation of z_i such that this will yield a large positive value of z_i when p_i is small. The combined p value can then be calculated with $p_c = 1 - \Phi(z)$.

Inverse chi-square method

Analogous to Stouffer's "inverse-normal" method, in the inverse chi-square method (not to be confused with Fisher's method) we transform the p values using the inverse of the cumulative distribution function of a chi-square distribution with one degree of freedom, which we denote by $F^{-1}(\cdot, 1)$. Hence

$$X^2 = \sum_{i=1}^{k} F^{-1}(1 - p_i, 1) \tag{3}$$

follows a chi-square distribution with $\text{df} = k$ and the combined p value is then given by $p_c = 1 - F(X^2, k)$.

Binomial test

The binomial test (Wilkinson 1951) follows naturally after noting that rejection versus non-rejection of a true null hypothesis can be considered to be a Bernoulli distributed random variable with the rejection probability denoted by α. Hence, when all k null hypotheses are true, the number of tests that lead to rejection, say r, follows a Binomial distribution, that is, $r \sim \text{Binomial}(k, \alpha)$. Therefore, we can compute the combined p value with

$$p_c = \sum_{x=r}^{k} \binom{k}{x} \alpha^x (1-\alpha)^{k-x}, \tag{4}$$

which gives the probability of obtaining r or more rejections ("successes") under the joint null. Hence, if $p_c \leq \alpha_c$, we reject the joint null. We can therefore regard the binomial test as a method for examining whether there is an "excess of significant results" among the set of k tests when assuming that the joint null hypothesis is true.

Note that α (i.e., the threshold for declaring an individual test as significant or not) can be different from α_c (the desired Type I error rate for the combined test). In fact, due to the discrete nature of p_c as computed with (4), the Type I error rate of the binomial test is bounded by α_c from above, that is, its actual Type I error rate may be lower than α_c (i.e., $\Pr(p_c \leq \alpha_c) \leq \alpha_c$ for the binomial test, with strict equality holding for the Fisher, Stouffer, and inverse chi-square methods).[1]

Bonferroni method

The last two methods to be described are more commonly thought of as multiple testing correction techniques, but are equally applicable in the present context. The first is the well-known Bonferroni correction (e.g., Goeman and Solari 2014), arguably one of the most commonly applied multiple testing correction techniques in research. As used for that purpose, the hypothesis-wise Type I error rate (i.e., α) is divided by the number of simultaneous tests (i.e., k). Then α/k is used as the significance threshold for the individual tests, which can be shown to control the family-wise Type I error rate (i.e., the probability of committing at least one Type I error across all k tests is then at most α). Equivalently, we can multiply the p value of each test by k and use α as the significance threshold (i.e., tests for which $p_i \times k \leq \alpha$ are significant).

Although the Bonferroni method is typically formulated as a multiple testing correction, it is also possible to use it as a method for combining p values. The motivation is that the joint null hypothesis should be rejected if at least one of the individual null hypotheses is rejected (cf. Simes 1986). For this, we only need to consider the *smallest* p value of the tests whose p values are to be combined. Therefore, the combined p value is then computed with

$$p_c = \min(1, \min(p_1, \dots, p_k) \times k),$$

which can be compared with α_c to test the joint null hypothesis.

In contrast to the previous methods (and contrary to common belief), the Bonferroni method does not assume independence among the tests (Goeman and Solari 2014). Hence, it controls the Type I error rate (in the sense that $\Pr(p_c \leq \alpha_c) \leq \alpha_c$ under the joint null) whether the tests are independent or not. However, the method is slightly conservative under independence (e.g., for $\alpha_c = 0.05$, the actual Type I error rate converges to $\lim_{k \to \infty} 1 - (1 - 0.05/k)^k = 1 - e^{-0.05} \approx 0.0488$) and it can be much more conservative when the tests are dependent.

The use of the Bonferroni correction as a method for combining p values shows that all multiple testing correction methods (that directly operate on the p values) can also be used as methods for combining p values. As long as one of the corrected p values leads to a rejection, the joint null can also be rejected. Therefore, one could consider other multiple testing correction methods that are less conservative than the Bonferroni method. The one

[1] Also note that when $\alpha_c < \alpha^k$, we can never reject the joint null with the binomial test. This may be a non-issue when testing a single joint null hypothesis, but in some situations (e.g., genetics), we often test many joint null hypotheses (e.g., for multiple genes), in which case α_c may be set to a low value to correct for multiple testing. It may then be impossible to reject a particular joint null hypothesis for which k is too low.

that immediately comes to mind is Holm's method (Holm 1979), which is also valid under any type of dependence structure among the tests and is less conservative than the Bonferroni correction (Goeman and Solari 2014).

Holm's procedure is a sequential "step-down" variant of the Bonferroni correction. It starts by multiplying the smallest p value with the number of simultaneous tests. If this corrected p value is larger than α, then none of the null hypotheses are rejected, whereas testing continues with the second smallest p value if the first step does lead to a rejection (and so on). As a method for combining p values (i.e., for testing the joint null hypothesis), Holm's procedure is therefore identical to the Bonferroni method, since only the smallest p value (multiplied by k) needs to be considered.

Other multiple testing correction methods such as those described by Simes (1986), Hochberg (1988), Hommel (1988), Benjamini and Hochberg (1995), and Benjamini and Yekutieli (2001) can also be used as methods for combining p values in the same manner and, compared to the Bonferroni/Holm method, can potentially lead to different conclusions when testing a joint null hypothesis. However, the methods by Benjamini and Hochberg (1995) and Benjamini and Yekutieli (2001) are not meant to control the family-wise Type I error rate, but the false discovery rate, and the other methods will only control the family-wise Type I error rate under independence or, in some cases, under the assumption of "positive dependence through stochastic ordering" (PDS, Goeman and Solari 2014).

Tippett's method

Tippett's method (also known as the Dunn-Šidák correction, based on Dunn 1958 and Šidák 1967) is another multiple testing correction (Tippett 1931). As used for this purpose, the hypothesis-wise error rate is set to $1 - (1 - \alpha)^{1/k}$, which controls the family-wise Type I error rate at α (i.e., it does not exhibit the slightly conservative behavior of the Bonferroni method), but is only guaranteed to do so under independence. The joint null hypothesis would therefore be rejected if $\min(p_i) \leq 1-(1-\alpha)^{1/k}$. Correspondingly, the combined p value based on Tippett's method is given by

$$p_c = 1 - (1 - \min(p_1, \ldots, p_k))^k,$$

which should be compared with α_c to test the significance of the joint null hypothesis.

2.2. Adjustments

In the previous section, we described six base methods for combining p values that are available in the **poolr** package. Five of these methods are only guaranteed to control the Type I error rate when the p values to be combined are independent. The Bonferroni correction is the only exception to this, but its Type I error rate can be quite conservative when the tests are dependent. Therefore, for all methods, we can apply certain adjustments that may help to bring their Type I error rates closer to the nominal level even when the tests are dependent.

For these adjustments, we assume that the p values to be combined are derived from k hypothesis tests with test statistics $t_1, \ldots, t_k$ that follow, under the joint null hypothesis, a multivariate normal distribution with means equal to zero, variances equal to unity, and correlation matrix R_t. For example, suppose we conduct k one-sample z tests of the null hypotheses $H_{0i}\colon \mu_i = \mu_{0i}$ for $i = 1, \ldots, k$ based on n observations of the random variables

$X_1, X_2, \ldots, X_k$, which themselves are multivariate normal. Then the scenario outlined above is exactly fulfilled since the joint null distribution of the test statistics is then also multivariate normal. Moreover, the correlations among the k random variables to be tested are exactly identical to the correlations among the test statistics of the z tests.

The same scenario arises in other contexts, at least asymptotically. For example, when conducting one- or two-sample t tests with interchanging variables, the test statistics converge in distribution to multivariate normality. The same applies when fitting k regression models with interchanging variables (each testing the association between some predictor and one of k different response variables, or vice-versa, when testing the association between one of k different predictors and some response variable). In fact, under the regularity conditions of the multivariate central limit theorem (e.g., Van der Vaart 1998), the interchanging variables do not have to follow a multivariate normal distribution for the test statistics to converge to multivariate normality. Hence, the scenario covers not only linear but also other types of regression models (e.g., logistic, Poisson). Moreover, while R_t is not exactly known in any of these examples, the correlations among the interchanging variables can be collected in a $k \times k$ correlation matrix denoted by R_x, which is a consistent estimate of R_t (in which case the multidimensional central limit theorem still holds; Härdle and Simar 2015).

Hence, in what follows, we assume that the dependence among the tests and hence p values arises through the correlations among a set of interchanging elements across the analyses, as reflected by the correlation matrix R_x. The latter can be computed from the data at hand or from reference databases in some fields (Liu *et al.* 2010).

Effective number of tests

When conducting k tests that are dependent in the manner described above, we can think of this as a scenario where we are effectively conducting a smaller number of independent tests. For example, suppose three SNPs are tested for their association with some phenotype of interest. If the SNPs are in perfect linkage equilibrium (and hence there is no correlation among the genotypes at the three locations), we are conducting three independent tests. On the other hand, if the SNPs are in perfect LD, then we are actually conducting the same test three times, leading to three identical p values, and hence the effective number of tests is one. Depending on the degree of LD among the SNPs, the effective number of tests will therefore lie somewhere between these two extremes. A variety of methods have been proposed based on this concept to quantify the degree of dependence among multiple tests.

All of these methods start by applying PCA to R_x, the matrix with the correlations among the interchanging elements (Cheverud 2001; Galwey 2009; Gao *et al.* 2008; Li and Ji 2005; Nyholt 2004). Let λ_i denote the i-th eigenvalue thereof such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k$. Then Cheverud (2001) and Nyholt (2004) propose to estimate the effective number of tests with

$$m_{CN} = 1 + (k-1)\left(1 - \frac{\mathrm{Var}(\lambda)}{k}\right), \tag{5}$$

where $\mathrm{Var}(\lambda)$ is the observed sample variance among the k eigenvalues. Alternatively, Li and Ji (2005) suggest to calculate the effective number of tests with

$$m_{LJ} = \sum_{i=1}^{k} h(|\lambda_i|), \tag{6}$$

where $h(x) = I(x \geq 1) + (x - \lfloor x \rfloor)$ and $\lfloor \cdot \rfloor$ is the floor function. Another approach, described by Gao *et al.* (2008), estimates the effective number of tests with

$$m_{GAO} = \underset{x \in \{1,\ldots,k\}}{\operatorname{argmin}} \left(\frac{\sum_{i=1}^{x} \lambda_i}{\sum_{i=1}^{k} \lambda_i} > C \right), \tag{7}$$

where C is a user-defined parameter which is usually set to 0.995 (i.e., m_{GAO} is the number of principal components needed such that $C \times 100\%$ of the total variance is accounted for). Finally, according to Galwey (2009), the effective number of tests should be computed with

$$m_{GAL} = \frac{\left(\sum_{i=1}^{k} \sqrt{\lambda_i'} \right)^2}{\sum_{i=1}^{k} \lambda_i'}, \tag{8}$$

where $\lambda_i' = \max[0, \lambda_i]$.

Although on first sight the calculations appear to be rather different, the methods have some common properties. When R_x is an identity matrix, then $\lambda_i = 1$ for $i = 1, \ldots, k$ and hence $m = k$ (where m denotes one of the estimates of the effective number of tests described above). The only exception to this is m_{GAO}, as it may yield an estimate less than k, depending on C and k (e.g., when $C = 0.995$ and $k > 200$, then $m_{GAO} < k$). On the other hand, when all correlations in R_x are equal to 1, then $\lambda_1 = k$ and $\lambda_i = 0$ for $i = 2, \ldots, k$ and hence $m = 1$ for all methods. However, differences in the value of m can arise for intermediate cases, as we will illustrate later on.

Once m is calculated with one of the methods described above, the methods that were described in Section 2.1 can be adjusted based on this quantity. In particular, the test statistic for Fisher's method is then computed with

$$\tilde{X}^2 = \frac{m}{k} \times X^2, \tag{9}$$

where X^2 is given by (1). Then $\tilde{X}^2$ is assumed to follow a chi-square distribution with df $= 2m$ and hence the combined p value is now calculated with $p_c = 1 - F(\tilde{X}^2, 2m)$. Similarly, for the inverse chi-square method, we also use (9), except that X^2 is then given by (3) and the combined p value is calculated with $p_c = 1 - F(\tilde{X}^2, m)$. The test statistic for Stouffer's method is adjusted with

$$\tilde{z} = \sqrt{\frac{m}{k}} \times z,$$

where z is given by (2). The combined p value is then calculated with $p_c = 1 - \Phi(\tilde{z})$. For the binomial method, we first compute $\tilde{r} = \lfloor \frac{r \times m}{k} \rfloor$ and then obtain the combined p value with

$$p_c = \sum_{x=\tilde{r}}^{m} \binom{m}{x} \alpha^x (1-\alpha)^{m-x}.$$

Instead of using the floor function, one could also compute $\tilde{r}$ using rounding, but we prefer the former for its conservativeness. Finally, for the Bonferroni and Tippett methods, we simply replace k with m and hence the combined p values are calculated with

$$p_c = \min(1, \min(p_1, \ldots, p_k) \times m)$$

and

$$p_c = 1 - (1 - \min(p_1, \ldots, p_k))^m,$$

respectively.

The adjustments are made in such a way so that they simplify to the base methods when $m = k$. Moreover, when $m = 1$, the same test is effectively conducted k times, leading to k identical p values (i.e., $p \equiv p_1 = \ldots = p_k$). In this case, the "combined" p value based on the adjusted methods will be identical to p, except for the binomial method, which either yields a p value of 1 (when $p > \alpha$, so that $\tilde{r} = 0$) or equal to α (when $p \leq \alpha$, so that $\tilde{r} = 1$).

However, it should be noted that these adjustments based on an estimate of the effective number of tests are ad hoc rather than principled techniques. In fact, one can construct examples under which the assumed distributions of the adjusted test statistics do not correspond to their analytically provable distributions. Moreover, even estimates of the effective number of tests may not coincide with our expectations. For example, suppose that R_x is block-diagonal with b blocks of sizes $k_1, \ldots, k_b$ with perfect correlation within blocks, that is,

$$\mathrm{R}_x = \begin{bmatrix} 1_{k_1} & & & \\ & 1_{k_2} & & \\ & & \ddots & \\ & & & 1_{k_b} \end{bmatrix}$$

where 1_{k_j} is a $k_j \times k_j$ matrix of all 1s. Then the eigenvalues of R_x are simply the block sizes sorted by decreasing magnitude followed by $k - b$ zeros. In this case, $m_{LJ} = b$ as we would expect (i.e., the effective number of tests is then equal to the number of blocks), but this is not guaranteed for the other estimators.[2] Hence, adjustments based on an effective number of tests have received some criticism in the literature (e.g., Dudbridge and Koeleman 2004; Salyakina, Seaman, Browning, Dudbridge, and Muller-Myhsok 2005) and should therefore be applied with caution.

Correlation among dependent p values

Although PCA is typically applied to R_x, one can also make the argument that this matrix does not directly reflect the degree of dependence among the p values. In addition, one also needs to consider the type and sidedness of the tests conducted. For example, suppose for n observations of the random variables $X_1, X_2, \ldots, X_k$, we conduct k one-sample t tests of the null hypotheses $H_{0i}\colon \mu_i = \mu_{0i}$ for $i = 1, \ldots, k$. When conducting two-sided tests, then the correlations among the p values are roughly approximated by $|\mathrm{R}_x|^3$ under the joint null hypothesis, which we can see by means of a small simulation study:

```
R> set.seed(2468)
R> n <- 20
R> Rmat <- matrix(0.8, nrow = 2, ncol = 2)
R> diag(Rmat) <- 1
```

[2] When the blocks are of equal size, then both m_{LJ} and m_{GAL} are guaranteed to be equal to b, but still not so for m_{CN} and m_{GAO}.

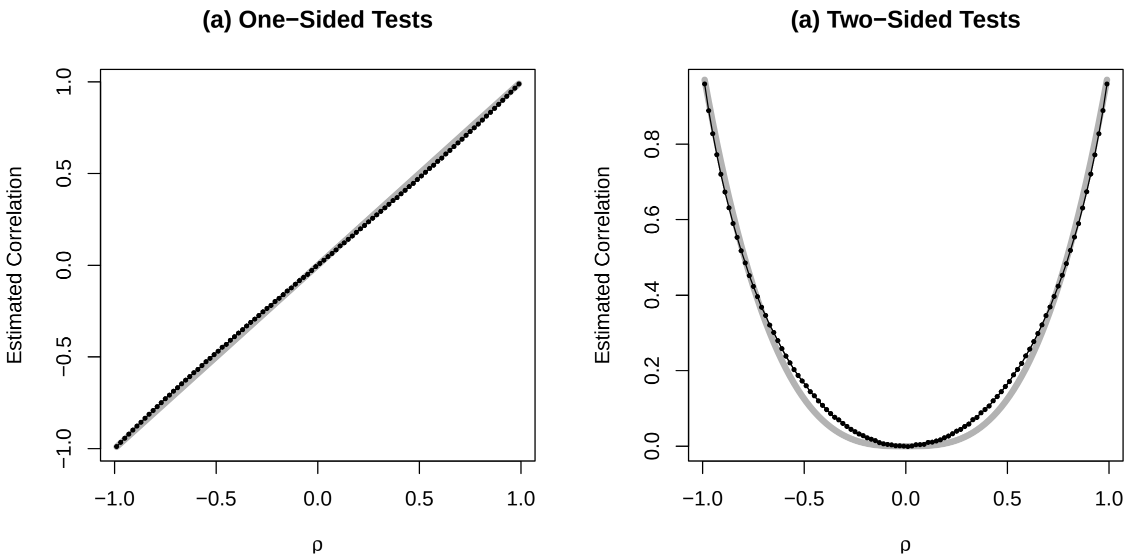

Figure 1: Estimated correlation between (a) one- and (b) two-sided p values from one-sample t tests as a function of the true correlation (ρ) between the two variables tested (the gray lines are based on the approximations ρ and $|\rho|^3$, respectively).

```
R> P <- replicate(1000, {
+    X <- mvtnorm::rmvnorm(n, mean = c(0, 0), sigma = Rmat)
+    p.two.sided.1 <- t.test(X[, 1], alternative = "two.sided")$p.value
+    p.two.sided.2 <- t.test(X[, 2], alternative = "two.sided")$p.value
+    p.one.sided.1 <- t.test(X[, 1], alternative = "greater")$p.value
+    p.one.sided.2 <- t.test(X[, 2], alternative = "greater")$p.value
+    c(p.two.sided.1, p.two.sided.2, p.one.sided.1, p.one.sided.2)})
R> Rmat[1, 2]^3

[1] 0.512

R> cor(P[1, ], P[2, ])

[1] 0.5174277
```

On the other hand, for one-sided tests, the correlations among the p values are approximately given by R_x, as we can see in this example:

```
R> Rmat[1, 2]

[1] 0.8

R> cor(P[3, ], P[4, ])

[1] 0.7889211
```

Figure 1 shows that these approximations roughly hold for values of ρ between -0.99 and 0.99. The same approximations also apply when conducting two-sample t tests for a set of k variables and for regression models (either with a single response variable and k different predictors or vice-versa).

One could easily obtain even better approximations using more complex polynomials, but under the assumptions stated earlier, we can derive the correlations among the p values directly. In particular, assume $[t_i, t_j]^\top \sim \mathrm{MVN}\left([0,0]^\top, \left[\begin{smallmatrix} 1 & \rho_{ij} \\ \rho_{ij} & 1 \end{smallmatrix}\right]\right)$, where ρ_{ij} is the element in the i-th row and j-th column of R_t. Making use of the "law of the unconscious statistician" (DeGroot and Schervish 2011), the correlation among the one-sided p values, $p_i = 1 - \Phi(t_i)$ and $p_j = 1 - \Phi(t_j)$, is then given by

$$\mathrm{Cor}(p_i, p_j) = 12 \iint\limits_{-\infty}^{+\infty} (1 - \Phi(t_i)) \times (1 - \Phi(t_j)) f(t_i, t_j)\, dt_i\, dt_j - 3, \tag{10}$$

since $\mathrm{E}[p_i] = \mathrm{E}[p_j] = 1/2$ and $\mathrm{Var}[p_i] = \mathrm{Var}[p_j] = 1/12$. For two-sided p values, $p_i = 2(1 - \Phi(|t_i|))$ and $p_j = 2(1 - \Phi(|t_j|))$, the correlation is given by

$$\mathrm{Cor}(p_i, p_j) = 12 \iint\limits_{-\infty}^{+\infty} 2(1 - \Phi(|t_i|)) \times 2(1 - \Phi(|t_j|)) f(t_i, t_j)\, dt_i\, dt_j - 3. \tag{11}$$

For example, if $\rho_{ij} \in (0, 0.3, 0.6, 0.9, 0.95)$, then $\mathrm{Cor}(p_i, p_j)$ is equal to (0.000, 0.288, 0.582, 0.892, 0.946) for one-sided and (0.000, 0.056, 0.250, 0.701, 0.831) for two-sided tests.

Hence, it may be advantageous to construct a matrix based on the correlations among the p values using these "exact" computations before computing the effective number of tests as described above. For this purpose, **poolr** provides appropriate lookup tables for (10) and (11) that obviate the need for the numerical integration described above (which can be time consuming when k is large, since $k(k-1)/2$ correlations need to be computed).

Empirically-derived null distributions using pseudo replicates

Another approach to account for the dependence among the p values is to make use of resampling methods such as permutation tests (e.g., Good 2013; Westfall and Young 1993). In fact, in certain research areas (e.g., genomics), permutation tests are often regarded as a gold-standard technique (Johnson *et al.* 2010; Moskvina *et al.* 2011).

The basic idea behind a permutation test is to reshuffle the data in such a way that relevant features of the data structure are preserved (e.g., the correlation among multiple predictors or response variables) except for the association to be tested. Based on the reshuffled data, we then compute the test statistic of interest. By repeating this process a large number of times, an empirical distribution of the test statistic under the null hypothesis is generated. The proportion of values in this empirical distribution that are at least as extreme as the observed test statistic (i.e., the one obtained with the original data) then yields the p value of the permutation test.

As a simple example of how permutation tests can be used to combine p values, suppose we test the association between the case-control status of a group of subjects with each of k SNPs in a gene using the Cochran-Armitage trend test (e.g., Laird and Lange 2011; Ziegler and König

2010) with the goal of testing the significance of the gene as a whole. The smallest p value among the k SNPs could then serve as the test statistic of interest. For the permutation test, we randomly reshuffle the case-control status of the subjects (preserving the degree of LD among the SNPs), computing the smallest p value of the trend tests for each rearrangement. The permutation-based combined p value of the k tests is then given by the proportion of times that the smallest p value under the permutation distribution is equal to or smaller than the smallest observed p value (i.e., closer to 0 is more "extreme" for this test statistic).

Using the smallest p value as the test statistic of interest is a common approach in permutation testing (e.g., Laird and Lange 2011; Ziegler and König 2010). It is easy to see that this is equivalent to using either the Bonferroni or Tippett's method as the test statistic of interest (for the permutation test, the two methods yield an identical combined p value, since they are both monotonic functions of $\min(p_1, \ldots, p_k)$). However, since no single method for combining p values is optimal (i.e., none of the methods is a uniformly most powerful test; Birnbaum 1954), one could also consider using the combined p value from one of the other base methods as the test statistic of interest (e.g., for GWAS, Moskvina *et al.* 2011 actually recommend to base the permutation procedure on Fisher's method).

Permutation-based methods are attractive, as they can be regarded as non-parametric, valid under broad assumptions, and in some sense "exact" (Ernst 2004). If implemented correctly, the test also automatically reflects relevant properties of the data such as the correlations among the predictors or response variables (Conneely and Boehnke 2007). An important drawback is the computational burden of the method. An exact permutation test requires computing the test statistic of interest under every possible rearrangement of the data. However, the number of rearrangement is often so large as to make this computationally infeasible. Instead, a large number of random rearrangements are typically used. Regardless, to obtain a stable approximation of the p value that would have been obtained with an exact permutation test, we need to generate the test statistic under a large number of random rearrangements, which can still be a substantial computational burden (Moskvina *et al.* 2011), especially when permutation tests need to be repeatedly conducted. For example, in GWAS, one might easily test 20,000 or more genes in this manner. Unless heavily parallelized, these computations can take days or even weeks to complete. Moreover, since the p values for the genes would usually be corrected for multiple testing, it is necessary to estimate tail-area p values accurately, possibly requiring 10^6–10^7 iterations per gene (e.g., using a Bonferroni correction for multiple testing in 20,000 genes, the difference between $.00001 \times 20{,}000 = 0.2$ and $.000001 \times 20{,}000 = 0.02$ is highly relevant).

We can greatly speed up this process by using an empirically-derived null distribution that is not based on the original data, but by using pseudo replicates (e.g., Lin 2005; Liu *et al.* 2010). The general logic of this procedure is as follows. As described earlier, we assume that the joint null distribution of the test statistics of the k tests that were conducted can be approximated by a multivariate normal distribution with a mean vector of zeros and that R_x (the matrix with the correlations among the interchanging elements across the tests) serves as an estimate of the variance-covariance matrix of these test statistics. Hence, let Z denote an $(s \times k)$ matrix of s replicates of random data generated from such a k-dimensional multivariate normal distribution and either $\mathrm{P} = 1 - \Phi(\mathrm{Z})$ or $\mathrm{P} = 2 \times (1 - \Phi(|\mathrm{Z}|))$ – depending on whether one- or two-sided tests were originally conducted – the matrix with the corresponding p values, with $\Phi(\cdot)$ applied element-wise. An empirically-derived null distribution can then be obtained by applying a particular base method to the values in each row of P, yielding $p_c^1, \ldots, p_c^s$. The combined p value is then given by

$$p_c = \frac{\sum_{l=1}^{s} \mathrm{I}(p_c^l \leq p_c^{\text{obs}}) + 1}{s+1}, \tag{12}$$

where p_c^{obs} is the combined p value obtained from the observed p values and $\mathrm{I}(\cdot)$ is the indicator function. We add 1 in the numerator and denominator to avoid the possibility of obtaining a combined p value equal to zero and to yield a test with the correct size (Phipson and Smyth 2010).

In comparison to a "proper" permutation test, this approach requires only a fraction of the computation time. A further advantage of this method is that it does not require access to the raw data, only the p values of the individual tests and the correlation matrix R_x (and in some situations, the latter can be approximated based on reference data sets or other external information).

Generalized methods derived under dependence

Finally, it is also possible to modify the Fisher, Stouffer, and inverse chi-square methods so that the dependence structure is directly considered in the construction of the respective test statistic and the calculation of the combined p value. The first method is based on Brown (1975), who described a modification of Fisher's method for pooling the results of k one-sided one-sample z tests of the null hypotheses $H_{0i}\colon \mu_i \leq \mu_{0i}$ for the random variables $X_1, X_2, \ldots, X_k$ that are assumed to have a joint multivariate normal distribution. This is in fact an example of the scenario outlined earlier, since the joint null distribution of the test statistics is then also multivariate normal and the correlations among the k random variables to be tested are exactly identical to the correlations among the test statistics of the z tests (i.e., $\mathrm{R}_x = \mathrm{R}_t$).

Under the joint null hypothesis, the X^2 value of Fisher's method then has expected value $E(X^2) = 2k$ and variance $\mathrm{Var}(X^2) = 4k + 2\sum_{i=1}^{k-1}\sum_{j>i}^{k} \mathrm{Cov}(-2\ln(p_i), -2\ln(p_j))$, where the covariance term is given by

$$\mathrm{Cov}(-2\ln(p_i), -2\ln(p_j)) = 4 \iint\limits_{-\infty}^{+\infty} \ln(1-\Phi(t_i)) \times \ln(1-\Phi(t_j)) f(t_i, t_j)\, dt_i\, dt_j - 4, \tag{13}$$

which can be computed using numerical integration under the stated assumptions. Brown (1975) then proposed to use a Satterthwaite approximation to the distribution of X^2 based on a scaled chi-square distribution, that is, assume that $X^2 \sim c\chi^2_f$ (or equivalently, $X^2/c \sim \chi^2_f$), where χ^2_f is a chi-square random variable with df $= f$. In that case, X^2 has expected value cf and variance $2fc^2$ and we can equate these quantities to $E(X^2)$ and $\mathrm{Var}(X^2)$, respectively. Solving these equations for c and f yields $f = 2(\mathrm{E}(X^2))^2/\mathrm{Var}(X^2)$ and $c = \mathrm{Var}(X^2)/(2\mathrm{E}(X^2))$. Following this, the combined p value can then be calculated with $p_c = 1 - F(X^2/c, f)$.

As noted by Yang, Li, Williams, and Buu (2016), for two-sided p values, the only change required is that we compute the covariance with

$$\mathrm{Cov}(-2\ln(p_i), -2\ln(p_j)) = 4 \iint\limits_{-\infty}^{+\infty} \ln(2(1-\Phi(|t_i|))) \times \ln(2(1-\Phi(|t_j|))) f(t_i, t_j)\, dt_i\, dt_j - 4. \tag{14}$$

The rest of the procedure remains unchanged. While Brown (1975), Kost and McDermott (2002), and Yang *et al.* (2016) have provided closed-form approximations to (13) and (14) to avoid the numerical integration step, the **poolr** package makes use of lookup tables for (13) and (14) that were precomputed using Gaussian quadrature to high precision (see below for details). In this way, we provide an "exact"' implementation of Brown's method that is computationally efficient.

The next base method whose test statistic can be directly calculated under dependence is Stouffer's method, which was generalized to dependent tests by Strube (1985). For this, we compute the combined test statistic with

$$z = \frac{\sum_{i=1}^{k} z_i}{\sqrt{\text{Var}(\sum_{i=1}^{k} z_i)}} \tag{15}$$

and $p_c = 1 - \Phi(z)$ as before. The difficulty then lies in computing the denominator, since $\text{Var}(\sum_{i=1}^{k} z_i) = k + 2\sum_{i=1}^{k-1}\sum_{j>i}^{k} \text{Cov}(\Phi^{-1}(1-p_i), \Phi^{-1}(1-p_j))$. For one-sided tests, Strube (1985) suggested to approximate $\text{Cov}(\Phi^{-1}(1-p_i), \Phi^{-1}(1-p_j))$ by ρ_{ij}, which follows directly from the scenario outlined earlier, since

$$\begin{aligned}&\text{Cov}(\Phi^{-1}(1-p_i), \Phi^{-1}(1-p_j)) = \\ &\quad \iint\limits_{-\infty}^{+\infty} \Phi^{-1}(1-(1-\Phi(t_i))) \times \Phi^{-1}(1-(1-\Phi(t_j))) f(t_i,t_j)\, dt_i\, dt_j = \rho_{ij}.\end{aligned} \tag{16}$$

For two-sided tests, the covariance is then given by

$$\begin{aligned}&\text{Cov}(\Phi^{-1}(1-p_i), \Phi^{-1}(1-p_j)) = \\ &\quad \iint\limits_{-\infty}^{+\infty} \Phi^{-1}(1-2(1-\Phi(|t_i|))) \times \Phi^{-1}(1-2(1-\Phi(|t_j|))) f(t_i,t_j)\, dt_i\, dt_j,\end{aligned} \tag{17}$$

which can be evaluated using numerical integration.

However, two aspects of Strube's method when applied to two-sided tests require further attention. First, (17) is an improper integral with an infinite discontinuity when either t_i or t_j is equal to 0, which raises the concern that (17) is divergent. Although we cannot prove this analytically, we can however demonstrate via simulation methods that (17) converges to a finite and fixed value and hence (17) must be convergent.[3] Second, Strube's method makes the implicit assumption that the z_i values are multivariate normal, so that the numerator in (15) is also normal. For one-sided tests, this is unobjectionable, at least for the assumed scenario where the test statistics are multivariate normal (in which case the transformation of the one-sided p values to the z_i values simply yields back the original test statistics). However, for two-sided tests, the z_i values are not multivariate normal (although each z_i is marginally normal), even when the test statistics were so to begin with. As a result, the numerator of (15) and hence z is not normal. Asymptotically (i.e., for sufficient large k), normality of z may

[3] For this, we generate a large number of random (t_i, t_j) pairs from a bivariate normal distribution with known correlation, transform these values into z_i and z_j, and then compute their covariance. For an increasing number of samples, the covariance between z_i and z_j converges to a finite and fixed value that is identical to the covariance derived via numerical integration of (17).

still hold, but demonstrating this for particular cases requires further assumptions underlying versions of the central limit theorem for dependent random variables (e.g., Billingsley 1995; Resnick 2014). For example, for tests based on data with a spatial or temporal configuration, this would require that the degree of dependence decays with the distance between the data points at a sufficiently fast rate. However, characterizing the specific conditions under which the asymptotic normality of z can be safely assumed would require further work. For these reasons, we would caution against the use of Strube's method for combining the results of two-sided tests at the moment.

As an alternative, one can generalize the inverse chi-square method in an analogous manner as was done by Brown for Fisher's method. In particular, for the test statistic (3), it generally holds that $E(X^2) = k$ and $\mathrm{Var}(X^2) = 2k + 2\sum_{i=1}^{k-1}\sum_{j>i}^{k} \mathrm{Cov}(F^{-1}(1-p_i,1), F^{-1}(1-p_j,1))$ under the joint null hypothesis. For one- and two-sided tests, the covariance is given by

$$\mathrm{Cov}(F^{-1}(1-p_i,1), F^{-1}(1-p_j,1)) = \\ \iint\limits_{-\infty}^{+\infty} F^{-1}(1-(1-\Phi(t_i)),1) \times F^{-1}(1-(1-\Phi(t_j)),1) f(t_i,t_j)\,dt_i\,dt_j - 1 \tag{18}$$

and

$$\mathrm{Cov}(F^{-1}(1-p_i,1), F^{-1}(1-p_j,1)) = \\ \iint\limits_{-\infty}^{+\infty} F^{-1}(1-2(1-\Phi(|t_i|)),1) \times F^{-1}(1-2(1-\Phi(|t_j|)),1) f(t_i,t_j)\,dt_i\,dt_j - 1, \tag{19}$$

respectively, which we can again evaluate numerically under the assumption of multivariate normality of the original test statistics. As in Brown's method, we use a Satterthwaite approximation with $f = 2(\mathrm{E}(X^2))^2/\mathrm{Var}(X^2)$ and $c = \mathrm{Var}(X^2)/(2\mathrm{E}(X^2))$ and then compute the combined p value with $p_c = 1 - F(X^2/c, f)$.

3. Implementation

This section describes the functions in the **poolr** package (Cinar and Viechtbauer 2022) which can be used to apply the methods presented in the previous section. The package is available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/package=poolr and can be installed and loaded in the usual manner:[4]

```
R> install.packages("poolr")
R> library("poolr")
```

The **poolr** package is primarily used via six functions that implement the "base methods" described in Section 2.1, namely:

- `fisher()`: for Fisher's method,
- `stouffer()`: for Stouffer's method,

[4] The development version of the package is hosted on GitHub (https://github.com/ozancinar/poolr).

- `invchisq()`: for the inverse chi-square method,
- `binomtest()`: for the binomial test,
- `bonferroni()`: for the Bonferroni method,
- `tippett()`: for Tippett's method.

The functions feature a consistent syntax. The following are the arguments of the `fisher()` function as an illustration.

```
R> args(fisher)

function (p, adjust = "none", R, m, size = 10000, threshold,
    side = 2, batchsize, nearpd = TRUE, ...)
NULL
```

We will explain the purpose of the various arguments in the following sections.

3.1. Combining independent p values

Independent p values can be combined by supplying a vector of p values to the `p` argument of the base functions. For example:

```
R> pvals <- c(0.02, 0.03, 0.08, 0.20)
R> (res <- fisher(pvals))

combined p-values with:      Fisher's method
number of p-values combined: 4
test statistic:              23.107 ~ chi-square(df = 8)
adjustment:                  none
combined p-value:            0.003228942
```

The results are stored as a list in an object of class '`poolr`' that is printed with the S3 method `print.poolr()`. The output displays the combined p value along with additional information about the methods used. The information shown in the third line varies with respect to the base function. The `fisher()`, `stouffer()`, and `invchisq()` functions display the respective test statistic and its assumed distribution under the joint null from which the combined p value is derived, whereas the output of the `binomtest()` function shows the number of significant individual p values.[5] The remaining functions, `bonferroni()` and `tippett()`, simply indicate the minimum individual p value. If only the combined p value is of interest, it is stored as list element `p`. Hence, for these toy data, the combined p value for all six methods can be obtained with:

```
R> fun <- c("fisher", "stouffer", "invchisq", "binomtest",  "bonferroni",
+    "tippett")
R> round(sapply(fun, function(f) do.call(f, list(p = pvals))$p), digits = 5)
```

[5] For `binomtest()`, the default value of α (i.e., the significance level of the individual tests) is 0.05. A different value of α can be passed to the function via the `...` argument (e.g., `binomtest(p, alpha = 0.01)`).

```
    fisher   stouffer   invchisq  binomtest bonferroni    tippett
   0.00323    0.00100    0.00507    0.01402    0.08000    0.07763
```

As the example shows, conclusions can differ (e.g., for $\alpha_c = 0.05$) depending on the method used for combining the p values.

Note that one should not conclude based on this example that the Bonferroni and Tippett methods are always less powerful than the other methods. Among other things, the power of the methods depends on whether the "signal" is concentrated in one or a few tests or distributed across all or many of them. In the former case, the Bonferroni and Tippett methods will tend to be more powerful, which we can illustrate with a simple simulation where we repeatedly generate normally distributed test statistics with mean 0 for all but one of k tests, turn these test statistics into (two-sided) p values, apply each method, and obtain its empirical rejection rate:

```
R> k <- 10
R> iters <- 10000
R> set.seed(8780)
R> Z <- replicate(iters, rnorm(k, mean = c(3, rep(0, k - 1)), sd = 1))
R> P <- 2 * pnorm(abs(Z), lower.tail = FALSE)
R> sapply(fun, function(f) mean(apply(P, 2,
+      function(pvals) do.call(f, list(p = pvals))$p) <= 0.05))
```

```
    fisher   stouffer   invchisq  binomtest bonferroni    tippett
    0.4341     0.2183     0.4858     0.0605     0.5926     0.5966
```

On the other hand, when the null hypothesis is false for all of the tests, then Fisher's, Stouffer's, and the inverse chi-square method will tend to have higher power:

```
R> Z <- replicate(iters, rnorm(k, mean = rep(1, k), sd = 1))
R> P <- 2 * pnorm(abs(Z), lower.tail = FALSE)
R> sapply(fun, function(f) mean(apply(P, 2,
+      function(pvals) do.call(f, list(p = pvals))$p) <= 0.05))
```

```
    fisher   stouffer   invchisq  binomtest bonferroni    tippett
    0.5549     0.5114     0.5467     0.2393     0.3006     0.3053
```

3.2. Combining dependent p values

Under dependence, the base functions can be modified using the adjustment techniques described in Section 2.2. In the following sections, we will demonstrate the use of these techniques on a set of correlated p values generated below. We will assume that there are $k = 5$ correlated tests and the test statistics derived from these tests follow a multivariate normal distribution with zero means and unit variances (i.e., the joint null hypothesis is true) and that the test statistics have a constant correlation of 0.7.

```
R> Rmat <- matrix(0.7, nrow = 5, ncol = 5)
R> diag(Rmat) <- 1
R> Rmat
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]  1.0  0.7  0.7  0.7  0.7
[2,]  0.7  1.0  0.7  0.7  0.7
[3,]  0.7  0.7  1.0  0.7  0.7
[4,]  0.7  0.7  0.7  1.0  0.7
[5,]  0.7  0.7  0.7  0.7  1.0
```

We can then simulate the test statistics and corresponding (two-sided) p values with:

```
R> ti <- c(mvtnorm::rmvnorm(1, mean = rep(0, 5), sigma = Rmat))
R> round(ti, digits = 5)

[1] 1.11566 1.02047 0.59187 2.04723 2.08762

R> pvals <- 2 * pnorm(abs(ti), lower.tail = FALSE)
R> round(pvals, digits = 5)

[1] 0.26457 0.30750 0.55394 0.04064 0.03683
```

Coincidentally, when (incorrectly) assuming independence, all methods except the ones by Bonferroni and Tippett yield a significant combined p value for these data:

```
R> round(sapply(fun, function(f) do.call(f, list(p = pvals))$p), digits = 5)

    fisher   stouffer   invchisq  binomtest bonferroni    tippett
   0.03770    0.02142    0.04782    0.02259    0.18416    0.17109
```

Adjusting based on the effective number of tests

As a first type of adjustment, we can estimate the effective number of tests (m) and incorporate this information into the methods for combining the p values. For this, we set argument `adjust` to `"nyholt"`, `"liji"`, `"gao"`, or `"galwey"` corresponding to (5), (6), (7), and (8), respectively.[6] Argument `R` is then used to specify the correlation matrix of the test statistics. The estimate of m is provided in the output along with a reference to the article where the method was first described. Below is an example where we adjust Fisher's method using an estimate of the effective number of tests based on Li and Ji (2005):

```
R> fisher(pvals, adjust = "liji", R = Rmat)

combined p-values with:      Fisher's method
number of p-values combined: 5
test statistic:              11.525 ~ chi-square(df = 6)
adjustment:                  effective number of tests (m = 3; Li & Ji, 2005)
combined p-value:            0.0734
```

[6] When using the Gao adjustment, $C = 0.995$ by default, but a different value of C can be passed to the function via the `...` argument (e.g., `fisher(pvals, adjust = "gao", R = Rmat, C = 0.95)`).

The base functions employ the `meff()` function of the **poolr** package to estimate the effective number of tests during this process. This function can also be called directly if only the estimate of m is of interest. For example, we can quickly compare the estimates of m across the different estimation methods with:

```
R> methods <- c("nyholt", "liji", "gao", "galwey")
R> sapply(methods, function(method) meff(Rmat, method = method))

nyholt   liji    gao galwey
     3      3      5      3
```

All estimates agree here except for the method by Gao *et al.* (2008), which gives the somewhat surprising estimate of $m = k$.

For clarity, we should note that `Rmat` is treated here synonymously as the correlation matrix among the test statistics, R_t, and as the correlation matrix of the interchanging elements across the five tests, R_x (e.g., when running five regression models, each testing the association between some predictor of interest and one of five different response variables). Assuming that the size of the sample (based on which the test statistics were obtained) was sufficiently large, the distinction between these two correlation matrices is negligible. However, this will not always be true and we will consider in more detail the relationship between these two types of correlation matrices in the discussion section.

Furthermore, in the example above, we used the correlation matrix among the test statistics (R_t) for estimating m. Earlier, we discussed the possibility of using the correlations among the p values instead. For the multivariate normal case, the `mvnconv()` function can be used to obtain these correlations (based on Equations 10 and 11).

The function has the following syntax: `mvnconv(R, side = 2, target, cov2cor = FALSE)`. The correlation matrix of the test statistics is passed to the function via the `R` argument, while argument `side` specifies the sidedness of the tests (i.e., `1` or `2` for one- or two-sided tests, respectively). The `target` argument is a character string to specify for which target statistic the covariance matrix should be obtained, and logical argument `cov2cor` can be set to `TRUE` if the covariance matrix should be converted into the corresponding correlation matrix. For the present purposes, we would like to convert `Rmat` into the correlation matrix of the two-sided p values (based on Equation 11), so we set `target = "p"` and `cov2cor = TRUE`:

```
R> (Rpmat <- mvnconv(Rmat, target = "p", cov2cor = TRUE))

          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 1.0000000 0.3589436 0.3589436 0.3589436 0.3589436
[2,] 0.3589436 1.0000000 0.3589436 0.3589436 0.3589436
[3,] 0.3589436 0.3589436 1.0000000 0.3589436 0.3589436
[4,] 0.3589436 0.3589436 0.3589436 1.0000000 0.3589436
[5,] 0.3589436 0.3589436 0.3589436 0.3589436 1.0000000
```

The resulting correlation matrix can then be used in combination with any of the base functions and methods for estimating the effective number of tests. For example:

```
R> fisher(pvals, adjust = "liji", R = Rpmat)
```

```
combined p-values with:      Fisher's method
number of p-values combined: 5
test statistic:              15.367 ~ chi-square(df = 8)
adjustment:                  effective number of tests (m = 4; Li & Ji, 2005)
combined p-value:            0.0524
```

Finally, to provide additional flexibility, the base functions also have an argument called `m` which allows the user to specify an estimate of the effective number of tests (e.g., as obtained with some other method not implemented in the **poolr** package). Returning to the earlier example where we used `Rmat` directly as input, these two approaches are therefore synonymous (output omitted):

```
R> fisher(pvals, adjust = "liji", R = Rmat)
R> fisher(pvals, m = meff(Rmat, method = "liji"))
```

Adjusting based on empirically-derived null distributions

When setting `adjust = "empirical"`, the combined p value is obtained via empirically-derived null distributions as described earlier. Argument `R` is then used to specify the correlation matrix of the test statistics, while argument `size` specifies the size of the null distribution that should be generated (s = 10,000 by default). The latter is done by the `empirical()` function, which uses (a slightly simplified version of) `mvtnorm::rmvnorm()` to generate the $s \times k$ matrix of pseudo replicates of the test statistics, converts them to one- or two-sided p values (depending on the `side` argument, which is set to `2` by default), and then applies the respective base method to each row. For example, we can apply this approach to Fisher's method with:

```
R> fisher(pvals, adjust = "empirical", R = Rmat)

combined p-values with:      Fisher's method
number of p-values combined: 5
test statistic:              19.208 ~ chi-square(df = 10)
adjustment:                  empirical distribution (size = 10000)
combined p-value:            0.106 (95% CI: 0.0996, 0.112)
```

Under the assumption that the combined p value represents an estimate of the "true" combined p value that would have been obtained under an infinite number of replications, the function also provides a confidence interval for the combined p value based on the Clopper and Pearson (1934) method (using `stats::binom.test()` for the computations).

Although generating the null distribution using pseudo replicates is much faster than performing a "proper" permutation test using the raw data, it is still computationally more demanding than the other adjustment techniques. The main factor that affects the computation time is the size of the empirical distribution generated. To reduce the computational burden, Liu *et al.* (2010) proposed a stepwise algorithm that starts with a relatively small null distribution and only proceeds to generate a larger distribution when the combined p value is small. The **poolr** package also provides the possibility to use such a stepwise algorithm via the `size` and `threshold` arguments of the base functions. In particular, for $j = 1, \dots,$ `length(size)`,

1. estimate the combined p value based on `size[j]`,
2. if the combined p value is $\geq$ than `threshold[j]`, stop (and report the combined p value), otherwise go back to 1.

By setting `size` to a vector with increasing values (e.g., `size = c(1000, 100000, 10000000)`) and `threshold` to decreasing values (e.g., `threshold = c(0.10, 0.01, 0)`), one can quickly obtain a fairly accurate estimate of the combined p value if it is far from significant (e.g., $\geq$ .10), but hone in on a more accurate estimate that is closer to 0. Note that the last value of `threshold` should be 0 (and is forced to be inside of the functions), so that the algorithm is guaranteed to terminate (hence, one can also leave out the last value of threshold, so `threshold = c(0.10, 0.01)` would also work in the example above). For example:

```
R> fisher(pvals, adjust = "empirical", R = Rmat,
+    size = c(1000, 10000, 100000), threshold = c(0.10, 0.01))

combined p-values with:      Fisher's method
number of p-values combined: 5
test statistic:              19.208 ~ chi-square(df = 10)
adjustment:                  empirical distribution (size = 1000)
combined p-value:            0.116 (95% CI: 0.0967, 0.137)
```

Since the combined p value is just a bit larger than 0.10 (the first threshold) in this example, we have decreased the computational burden, since only 1000 values of the null distribution needed to be generated. While the difference may be hardly noticeable in the present case, when applying such methods repeatedly under circumstances where many combined p values are likely to be insignificant (e.g., for a large number of genes in the GWAS setting), this approach can help to reduce the total computation time considerably. For example:[7]

```
R> Z <- mvtnorm::rmvnorm(1000, mean = rep(0, 5), sigma = Rmat)
R> P <- 2 * pnorm(abs(Z), lower.tail = FALSE)
R> system.time(p <- apply(P, 1, function(pvals)
+    fisher(pvals, adjust = "empirical", R = Rmat)$p))

   user  system elapsed
 53.780   0.004  53.784

R> system.time(p <- apply(P, 1, function(pvals)
+    fisher(pvals, adjust = "empirical", R = Rmat,
+      size = c(1000, 10000, 100000), threshold = c(0.10, 0.01))$p))

   user  system elapsed
 17.198   0.051  17.251
```

Generating the null distribution requires simulating a `size` $\times\, k$ matrix of pseudo test statistics. By default, this is done in a single step, so this entire matrix needs to be held in memory

[7] All timings reported in this paper were obtained by running the code on an Intel Xeon E5-2630v4 processor.

temporarily. When `size` and k are very large, doing so can lead to memory allocation problems. In cases where this problem occurs, it can be avoided by dividing the process that generates the null distribution into multiple batches with smaller sizes, which can be done by setting the `batchsize` argument to some value smaller than `size`. Doing so reduces memory usage, but leads to an increase in the computation time:

```
R> system.time(p <- apply(P, 1, function(pvals)
+    fisher(pvals, adjust = "empirical", R = Rmat, batchsize = 100)$p))
```

```
   user  system elapsed
 63.160   0.016  63.177
```

Note that `size` does not need to be an even multiple of `batchsize` (the size of the last batch will be adjusted appropriately such that the null distribution will always be of the requested `size`).

Adjusting based on generalized methods

As discussed earlier, the Fisher, Stouffer, and inverse chi-square methods can also be directly generalized to account for dependence. To make use of these methods, we set `adjust = "generalized"` in the respective base functions. Moreover, R_t needs to be converted into a matrix that contains the covariances among the appropriate transformation of the test statistics before it can be passed to the functions via the `R` argument. In principle, this step requires repeatedly solving double integrals numerically, which is computationally very demanding. This problem can be easily sidestepped via a well-known memoization technique, namely the use of a lookup table that contains pre-computed values of these covariances for the various “target” statistics. The dataset `mvnlookup` in the **poolr** package contains such a table:

```
R> head(mvnlookup)
```

```
   rhos m2lp_1 m2lp_2   z_1    z_2 chisq1_1 chisq1_2    p_1    p_2
1 1.000 4.0000 4.0000 1.000 1.0000   2.0000   2.0000 0.0833 0.0833
2 0.999 3.9949 3.9908 0.999 0.9811   1.9971   1.9956 0.0832 0.0830
3 0.998 3.9901 3.9823 0.998 0.9735   1.9944   1.9915 0.0831 0.0826
4 0.997 3.9854 3.9738 0.997 0.9669   1.9917   1.9875 0.0831 0.0823
5 0.996 3.9806 3.9653 0.996 0.9610   1.9891   1.9836 0.0830 0.0819
6 0.995 3.9758 3.9568 0.995 0.9555   1.9864   1.9796 0.0829 0.0816
```

Columns `m2lp_1` and `m2lp_2` correspond to (13) and (14), columns `z_1` and `z_2` to (16) and (17), and columns `chisq_1` and `chisq_2` to (18) and (19), respectively. The last two columns are the covariances among one- and two-sided p values, which we already made use of earlier (or rather, the correlations).

The values in the lookup table were computed using Gauss-Legendre quadrature with the help of the **pracma** package (Borchers 2021). For this, we used a very fine grid with dimensions 1000×1000 on the interval $[-5, 5]$ for both test statistics. Total construction time of the lookup table was about 45 hours using 18 cores in parallel on a workstation with two Intel Xeon E5-2630v4 processors. Values were cross-checked using `adaptIntegrate()` from the **cubature** package (Narasimhan, Johnson, Hahn, Bouvier, and Kiêu 2021) and via simulations.

Instead of using the table directly, `mvnconv()` conveniently allows the transformation of an entire R_t matrix into the required covariance matrix. The R_t matrix is passed to the function via the `R` argument, `target` should be set to either `"m2lp"`, `"z"`, or `"chisq1"` for functions `fisher()`, `stouffer()`, and `invchisq()`, respectively, and argument `side` to 1 or 2 depending on the sidedness of the tests (with 2 being the default). For example:

```
R> mvnconv(Rmat, target = "m2lp")

       [,1]   [,2]   [,3]   [,4]   [,5]
[1,] 4.0000 1.9286 1.9286 1.9286 1.9286
[2,] 1.9286 4.0000 1.9286 1.9286 1.9286
[3,] 1.9286 1.9286 4.0000 1.9286 1.9286
[4,] 1.9286 1.9286 1.9286 4.0000 1.9286
[5,] 1.9286 1.9286 1.9286 1.9286 4.0000
```

For convenience (and to reduce potential usage errors), when `mvnconv()` is used within a call to a base function with `adjust = "generalized"`, the `target` argument is automatically set to the appropriate option. Hence, the following code will run the appropriate "generalized" version of each test:

```
R> fisher(pvals, adjust = "generalized", R = mvnconv(Rmat))

combined p-values with:      Fisher's method
number of p-values combined: 5
test statistic:              6.559 ~ chi-square(df = 3.415)
adjustment:                  Brown's method
combined p-value:            0.115

R> stouffer(pvals, adjust = "generalized", R = mvnconv(Rmat))

combined p-values with:      Stouffer's method
number of p-values combined: 5
test statistic:              1.283 ~ N(0,1)
adjustment:                  Strube's method
combined p-value:            0.0998

R> invchisq(pvals, adjust = "generalized", R = mvnconv(Rmat))

combined p-values with:      inverse chi-square method
number of p-values combined: 5
test statistic:              3.78 ~ chi-square(df = 1.69)
adjustment:                  Satterthwaite approximation
combined p-value:            0.116
```

Although the results from Strube's method appear sensible here, we reiterate our caution about the use of this method for two-sided tests.

Note that all of the methods implemented in the **poolr** package are equally applicable to one-sided tests. For example, suppose that the test statistics simulated earlier were used to test one-sided hypotheses such that only positive values could lead to rejection. Then the one-sided p values would be:

```
R> pvals <- pnorm(ti, lower.tail = FALSE)
R> round(pvals, digits = 5)

[1] 0.13228 0.15375 0.27697 0.02032 0.01842
```

Only two minor changes are now required to the code above. First, whenever `mvnconv()` is called, we need to set argument `side = 1`, so that the appropriate column from the lookup table is used for converting the correlations to the covariances. For example:

```
R> fisher(pvals, adjust = "generalized", R = mvnconv(Rmat, side = 1))

combined p-values with:      Fisher's method
number of p-values combined: 5
test statistic:              7.18 ~ chi-square(df = 2.747)
adjustment:                  Brown's method
combined p-value:            0.0546
```

Moreover, when using `adjust = "empirical"` in any of the base functions, the `side` argument needs to be set in the same manner, so that the pseudo replicates of the test statistics are also appropriately converted into one-sided p values. For example:

```
R> fisher(pvals, adjust = "empirical", R = Rmat, side = 1)

combined p-values with:      Fisher's method
number of p-values combined: 5
test statistic:              26.14 ~ chi-square(df = 10)
adjustment:                  empirical distribution (size = 10000)
combined p-value:            0.0539 (95% CI: 0.0495, 0.0585)
```

4. Example

The example above was based on simulated data. In this section, we illustrate the use of the **poolr** package using real data from a candidate-gene study by Van Assche *et al.* (2017) who examined the association between 4947 SNPs and depressive symptoms, as measured with the Center of Epidemiologic Studies Depression Scale (CES-D Scale), in a sample of 982 adolescents. Here, we focus on the 23 SNPs that are part of the glutamate-related *GRID2IP* gene and the 886 adolescents with complete data on these 23 SNPs.

Object `grid2ip.geno` is a data frame with the genotypes of the subjects for these 23 SNPs. For example, the genotypes of the first six subjects for the first five SNPs are:

```
R> grid2ip.geno[1:6, 1:5]

  rs10267908 rs112305062 rs117541653 rs11761490 rs11773436
1        T/T         G/G         G/G        G/G        G/G
2        A/T         G/G         G/G        G/G        G/G
3        A/A         A/G         G/G        G/G        A/G
4        A/T         G/G         G/G        G/G        G/G
5        A/T         G/G         G/G        G/G        G/G
7        A/T         G/G         G/G        G/G        A/G
```

The corresponding (log-transformed) CES-D values of these subjects are contained in the vector `grid2ip.pheno`. For the first six subjects, the values are:

```
R> head(grid2ip.pheno)

[1] 2.3025851 2.1972246 0.6931472 1.9459101 1.0986123 3.0910425
```

Using the **genetics** package (Warnes 2021), we can transform the genotypes into the number of minor alleles in each SNP with:

```
R> G <- as.data.frame(lapply(grid2ip.geno, function(snp)
+    genetics::genotype(snp)))
R> X <- as.data.frame(lapply(G, function(snp)
+    genetics::allele.count(snp)[, 2]))
R> X[1:6, 1:5]

  rs10267908 rs112305062 rs117541653 rs11761490 rs11773436
1          2           0           0          0          0
2          1           0           0          0          0
3          0           1           0          0          1
4          1           0           0          0          0
5          1           0           0          0          0
6          1           0           0          0          1
```

Finally, using the number of minor alleles as predictors, we can fit an additive model (Laird and Lange 2011) to each SNP, with the (log-transformed) CES-D values as response variable:

```
R> pvals <- sapply(X, function(x) coef(summary(lm(grid2ip.pheno ~ x)))[2, 4])
```

The p values of the first five SNPs are:

```
R> pvals[1:5]

 rs10267908 rs112305062 rs117541653  rs11761490  rs11773436
0.011366143 0.506359643 0.123029250 0.099923843 0.001687646
```

Based on the genotypes, the corresponding 23 × 23 LD correlation matrix can be constructed with:

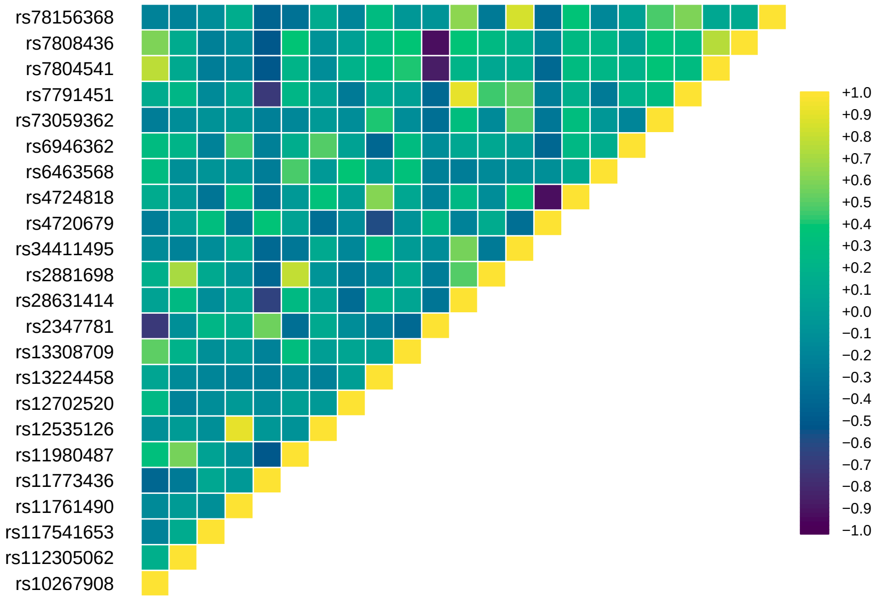

Figure 2: LD correlation matrix of 23 SNPs belonging to the *GRID2IP* gene.

```
R> LD <- genetics::LD(G)$r
R> LD[lower.tri(LD)] <- t(LD)[lower.tri(LD)]
R> diag(LD) <- 1
R> LD[1:5, 1:5]
```

```
             rs10267908  rs112305062 rs117541653   rs11761490  rs11773436
rs10267908    1.0000000  0.186527151 -0.19163219 -0.130103725 -0.38913793
rs112305062   0.1865272  1.000000000  0.14506274 -0.009624355 -0.26038439
rs117541653  -0.1916322  0.145062740  1.00000000 -0.096507476  0.09946650
rs11761490   -0.1301037 -0.009624355 -0.09650748  1.000000000 -0.01903163
rs11773436   -0.3891379 -0.260384395  0.09946650 -0.019031629  1.00000000
```

For convenience, the **poolr** package contains objects `grid2ip.p` and `grid2ip.ld` with the *p* values and the LD correlation matrix, respectively, so that the steps above could in principle be skipped. We can double-check that these objects are identical to the ones we generated above with:

```
R> c(all.equal(pvals, grid2ip.p), all.equal(LD, grid2ip.ld))
```

```
[1] TRUE TRUE
```

Figure 2 shows a heatmap of (the upper triangular part of) the LD correlation matrix.

Below we show how the combined *p* value for this gene can be obtained by using Fisher's method along with all possible adjustments (only one effective number of tests adjustment is shown for brevity; the estimate of m was 15 in this case). Note that the string passed to the `adjust` argument can also be abbreviated. Also, we use matrix `LD` (based on the interchanging elements) as an approximation to the correlations among the test statistics.

method	base	nyholt	liji	gao	galwey	empirical	generalized	permutation
"fisher"	8.857	7.849	6.160	7.175	5.479	3.012	3.581	3.062
"stouffer"	8.781	7.761	6.051	7.079	5.362	3.146	3.951	3.176
"invchisq"	8.352	7.415	5.845	6.789	5.213	2.942	3.420	2.981
"binomtest"	8.424	7.945	5.454	7.202	4.704	3.377		3.381
"bonferroni"	1.411	1.472	1.597	1.517	1.659	1.519		1.518
"tippett"	1.419	1.479	1.602	1.524	1.663	1.519		1.518

Table 1: Results ($-\log_{10}(p_c)$-transformed values) when applying all base methods and adjustments to the 23 SNPs in the *GRID2IP* gene.

```
R> c(fisher = fisher(pvals)$p,
+    liji = fisher(pvals, adjust = "liji", R = LD)$p,
+    emp = fisher(pvals, adjust = "emp", R = LD)$p,
+    brown = fisher(pvals, adjust = "gen", R = mvnconv(LD))$p)
```

```
      fisher         liji          emp        brown
1.389547e-09 6.918130e-07 7.999200e-04 2.622587e-04
```

All of the combined p values above agree that the gene is significantly associated with the phenotype of interest. For illustration purposes, we applied all base methods and their adjustments to these data, yielding the results shown in Table 1 (results are given as $-\log_{10}(p_c)$-transformed values for easier comparison of very small values). Regardless of the base and adjustment method used, all combined p values reached significance at $\alpha_c = 0.05$ (note: $-\log_{10}(0.05) \approx 1.301$). However, we see considerable differences in the level of significance depending on the method used.

In the last column of the table, we also provide the combined p values derived from conventional permutation tests using the raw data. For this, we tested the associations between the SNPs and the randomly reshuffled phenotype and then combined the resulting p values with each of the base methods. After repeating this process $s = 10^6$ times, we then computed the combined p value for each base method with (12). In this example, the approach using pseudo replicates (also using `size = 1e+06`) yielded almost identical results as the "proper" permutation tests. However, while the latter took approximately 7 hours to complete, the former took less than 10 seconds per base method.[8]

5. Comparison with other packages

Several R packages and functions are already currently available to combine (dependent) p values. In this section, we discuss their functionality and compare them to the **poolr** package based on the example data discussed in the previous section.

A first option is the `p.adjust()` function from the base package **stats**. This function provides a variety of methods for adjusting a set of p values for multiple testing. As described earlier,

[8] In the permutation test, we used `coef(summary(lm(...)))` to extract p values based on the permuted data in each iteration (similar to how we computed the observed p values for the 23 SNPs earlier). The code underlying these functions carries out various pre- and post-processing steps that are not relevant for the present purposes. We can reduce the computation time considerably by using `RcppEigen::fastLmPure()` (Bates and Eddelbuettel 2013). Doing so reduces the computation time to about 20 minutes.

Package	Fisher	Stouffer	Invchisq	Tippett	Logit
metap	8.857127	8.781088	8.351928	1.419033	8.724881
survcomp	8.857127	8.781088			8.724881
aggregation	8.857127		8.351928	1.419033	
gap	8.857127	8.781088			
poolr	8.857127	8.781088	8.351928	1.419033	

Table 2: Results ($-\log_{10}(p_c)$-transformed values) when applying various implementations of methods to the 23 SNPs in the *GRID2IP* gene that assume independence between the SNPs.

this function can therefore also be used for combining p values, although the Bonferroni correction and Holm's method are most relevant here, as they are the two methods available via this function that control the family-wise error rate under arbitrary assumptions. Using the example data, we can illustrate the equivalence of the Bonferroni and Holm methods and the implementation in the **poolr** package with:

```
R> c(p.adjust_bonferroni = min(p.adjust(pvals, method = "bonferroni")),
+    p.adjust_holm = min(p.adjust(pvals, method = "holm")),
+    poolr_bonferroni = bonferroni(pvals)$p)

p.adjust_bonferroni       p.adjust_holm    poolr_bonferroni
         0.03881585          0.03881585          0.03881585
```

For researchers that want to use methods that were specifically developed for combining p values, several packages are available (via CRAN and **Bioconductor**). The **metap** package (Dewey 2021) is the most comprehensive one, providing a wide variety of methods for combining independent p values, including the ones part of the **poolr** package except for the Bonferroni method. Furthermore, the `combine.test()` function from the **survcomp** package (Schroeder *et al.* 2011) provides implementations of the Fisher and Stouffer methods and the logit-transformation method by George (1977), whereas the **aggregation** package (Yi and Pachter 2018) provides functions for the Fisher, Tippett, and inverse chi-square methods (the latter being a special case of the method by Lancaster 1961). Finally, the **gap** package (Zhao 2007) provides functions for Fisher's and Stouffer's methods. Below, we show how to apply these methods to the example data.[9] The $-\log_{10}(p_c)$-transformed values provided in Table 2 demonstrate that results coincide across packages.

```
R> k <- length(pvals)
R> invisible(capture.output(pkgs_ind <- matrix(c(
+    metap::sumlog(pvals)$p,
+    metap::sumz(pvals)$p,
+    metap::invchisq(pvals, 1)$p,
+    metap::minimump(pvals)$p,
```

[9] Since `gap::metap()` always generates output via `cat()`, we use `invisible(capture.output(...))` to suppress it. Also, for Stouffer's method, the function incorrectly divides the provided p values by 2, so we must double them first to obtain the correct result (it also incorrectly takes the absolute value of (2) before computing p_c, which does not affect results in the present case since z is positive to begin with, but would have the effect of turning a set of really insignificant p values into a significant combined p value).

```
+    metap::logitp(pvals)$p,
+    survcomp::combine.test(pvals, method = "fisher"),
+    survcomp::combine.test(pvals, method = "z.transform"), NA, NA,
+    survcomp::combine.test(pvals, method = "logit"),
+    aggregation::fisher(pvals), NA,
+    aggregation::lancaster(pvals, rep(1, k)),
+    aggregation::sidak(pvals), NA,
+    gap::metap(data.frame(p = rbind(unname(pvals)),
+      n = rbind(rep(1, k))), N = k, prefixp = "p.", prefixn = "n.")$p,
+    gap::metap(data.frame(p = rbind(2 * unname(pvals)),
+      n = rbind(rep(1, k))), N = k, prefixp = "p.", prefixn = "n.")$p1,
+    NA, NA, NA,
+    poolr::fisher(pvals)$p,
+    poolr::stouffer(pvals)$p,
+    poolr::invchisq(pvals)$p,
+    poolr::tippett(pvals)$p, NA), nrow = 5, byrow = TRUE,
+    dimnames = list(c("metap", "survcomp", "aggregation", "gap", "poolr"),
+      c("Fisher", "Stouffer", "Invchisq", "Tippett", "Logit"))))
R> -log10(pkgs_ind)
```

These results ignore the fact that the 23 p values to be combined are dependent. For combining dependent p values, several other R packages are available. The **CombinePValue** package (Dai *et al.* 2014) provides an approach that is analogous to Brown's method, but (in addition to the actual p values to be combined) requires the user to provide a matrix of p values based on which the variance-covariance matrix of the $-2\ln(p_i)$ will be estimated. In other words, instead of using analytic methods such as (13) and (14) to construct this variance-covariance matrix, it is estimated empirically. As described in the documentation, the matrix of p values used for this step can be generated based on the raw data using resampling methods. Below, we construct such a matrix by repeatedly permuting the phenotype. Using the same variance-covariance matrix (after the $-2\ln()$ transformation) as input for `fisher()` yields identical results.[10]

```
R> P <- replicate(200, {
+    yperm <- sample(grid2ip.pheno)
+    sapply(X, function(x) coef(summary(lm(yperm ~ x)))[2, 4])})
R> CombinePValue::selfcontained.test(pvals, p_permu = P)[[1]]

[1] 6.693278e-05

R> fisher(pvals, adjust = "gen", R = cov(t(-2 * log(P))))$p

[1] 6.693278e-05
```

Ideally, one would want to increase the number of replicates, but doing so would be time intensive (e.g., for 10,000 replicates, the code above takes a bit over 4 minutes to complete,

[10] Very minor discrepancies can occur in some cases because **CombinePValue** constrains p values to fall in the interval $[10^{-6}, 1-10^{-6}]$, while no such restriction is applied in **poolr**.

while `fisher(pvals, adjust = "generalized", R = mvnconv(LD))` takes a fraction of a second). Moreover, if one generates the p value matrix by resampling the raw data, one could just as well apply a proper permutation test directly as part of this process (instead of using this step only for generating the matrix that serves as input for the `p_permu` argument).

As another alternative, the **EmpiricalBrownsMethod** package (Poole 2021) also implements a version of Brown's method and just like the **CombinePValue** package sidesteps the numerical integration but does so in a slightly different manner by requiring the user to input the raw data matrix that contains the interchanging elements. Each variable in this matrix is then transformed into percentiles using its empirical cumulative distribution function (as estimated via `stats::ecdf()`) which in turn are $-2\ln()$-transformed. These transformed values are then used to estimate the variance-covariance matrix of the $-2\ln(p_i)$ values. Using the example data, one would therefore use this package as follows:

```
R> EmpiricalBrownsMethod::empiricalBrownsMethod(t(X), p_values = pvals)

[1] 3.288796e-09
```

If we manually apply the step that estimates the variance-covariance matrix, replace the diagonal values by 4 (which we know to be true and is also done by **CombinePValue**), and then use this matrix as input to `fisher()`, we obtain the same result:

```
R> V <- EmpiricalBrownsMethod:::calculateCovariances(t(X))
R> diag(V) <- 4
R> fisher(pvals, adjust = "gen", R = V)$p

[1] 3.288796e-09
```

However, this result seems "too significant" compared to the combined p value we obtained above when using the **CombinePValue** approach or Brown's method proper (recall that `fisher(pvals, adjust = "gen", R = mvnconv(LD))$p` yields 0.000262). There are two reasons for this discrepancy. First, in the present context, the variables in `X` (i.e., the number of minor alleles in each SNP) can only take on three unique values and so will the $-2\ln()$-transformed percentiles based on which the covariances are estimated. Hence, each covariance is estimated based on a 3×3 contingency table where the marginal distributions of the two variables are also often quite unbalanced. As a result, the covariances in `V` are much smaller than what they should be, at least when compared to `cov(t(-2 * log(P)))` from above or when using the "exact" covariances from `mvnconv(LD, target = "m2lp")`. This leads to an undercorrection for dependence in Brown's method and hence to a combined p value that is anticonservative. A way around this problem might be to simulate a data matrix from a multivariate normal distribution based on the same variance-covariance matrix as the original data and use this as input for the method:[11]

```
R> S <- mvtnorm::rmvnorm(886, mean = rep(0, k), sigma = LD)
R> EmpiricalBrownsMethod::empiricalBrownsMethod(t(S), p_values = pvals)
```

[11] Note, however, that this approach is stochastic and will lead to (slightly) different results on repeated runs.

```
[1] 1.082403e-05
```

```
R> V <- EmpiricalBrownsMethod:::calculateCovariances(t(S))
R> diag(V) <- 4
R> fisher(pvals, adjust = "gen", R = V)$p
```

```
[1] 1.082403e-05
```

Although improved, this result still seems anticonservative. The reason for this is that the approach taken by `empiricalBrownsMethod()` implicitly assumes that the *p* values to be combined are one-sided. We can demonstrate this by comparing the variance-covariance matrix obtained above with the one we obtain using `mvnconv()` when setting `side = 1`:

```
R> round(unname(V), digits = 4)[1:4, 1:8]
```

```
        [,1]   [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8]
[1,]  4.0000 0.5769 -0.6079 -0.1057 -1.2269  1.1555 -0.0490  0.7288
[2,]  0.5769 4.0000  0.4935  0.0387 -0.9727  2.2010  0.0343 -0.6188
[3,] -0.6079 0.4935  4.0000 -0.2153  0.3087  0.2160 -0.2581 -0.3923
[4,] -0.1057 0.0387 -0.2153  4.0000 -0.1508 -0.2183  3.4120  0.0971
```

```
R> mvnconv(LD, target = "m2lp", side = 1)[1:4, 1:8]
```

```
        [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8]
[1,]  4.0000  0.6351 -0.6006 -0.4123 -1.1635  1.1999 -0.3036  0.8749
[2,]  0.6351  4.0000  0.4881 -0.0326 -0.8009  2.0748 -0.0098 -0.6245
[3,] -0.6006  0.4881  4.0000 -0.3099  0.3300  0.2184 -0.3380 -0.3752
[4,] -0.4123 -0.0326 -0.3099  4.0000 -0.0618 -0.3005  3.5402 -0.0876
```

Clearly, the former is an approximation to the latter, while for two-sided *p* values one should really use:

```
R> mvnconv(LD, target = "m2lp")[1:4, 1:8]
```

```
       [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]
[1,] 4.0000 0.1366 0.1440 0.0660 0.5922 0.4575 0.0352 0.2522
[2,] 0.1366 4.0000 0.0821 0.0004 0.2642 1.2526 0.0000 0.1563
[3,] 0.1440 0.0821 4.0000 0.0367 0.0383 0.0170 0.0439 0.0544
[4,] 0.0660 0.0004 0.0367 4.0000 0.0014 0.0345 3.2279 0.0028
```

or more directly:

```
R> fisher(pvals, adjust = "gen", R = mvnconv(LD))$p
```

```
[1] 0.0002622587
```

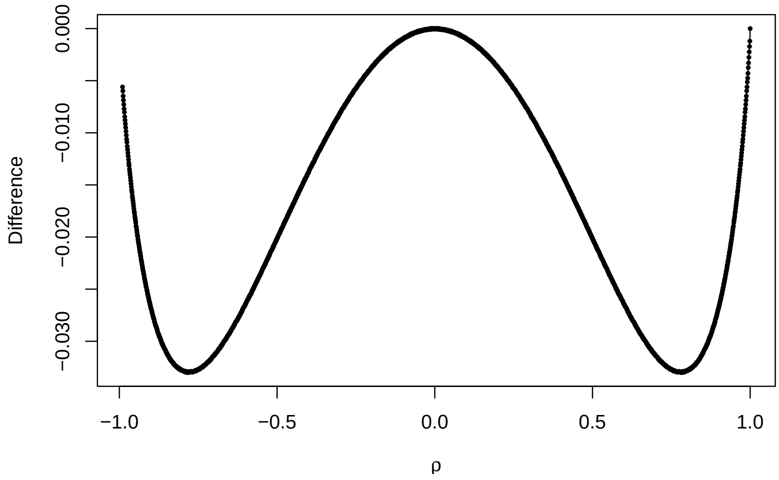

Figure 3: Difference between (14) and the covariances as approximated by the **TFisher** package as a function of ρ.

Hence, the **EmpiricalBrownsMethod** package, at least in its present form, should only be used for combining one-sided p values and care must be taken when the interchanging variables have rather discrete distributions (e.g., as when using genotype data in GWAS).

In contrast to the previous two options, the **TFisher** package (Zhang *et al.* 2020) provides an approximation to Brown's method that does not require access to the raw data. Although the manuscript does not explain how the covariances among the $-2\log(p_i)$ values are approximated, the source code indicates that the package simply computes $4 \times \mathrm{R}^2$ (with the squaring done element-wise). Using the same approximation as input for the `fisher()` function yields the same combined p value:

```
R> 1 - TFisher::p.tpm(sum(-2 * log(pvals)), n = k, tau1 = 1, M = LD)

[1] 0.0002822384

R> fisher(pvals, adjust = "gen", R = 4 * LD^2)$p

[1] 0.0002822384
```

This result is quite close to the combined p value obtained above with the "exact" version of Brown's method. The explanation for this is provided by Figure 3, which shows that the simple approximation for the covariances used by **TFisher** is surprisingly accurate, deviating at most by -0.033 from the covariances computed based on (14). However, here we now have the opposite problem compared to **EmpiricalBrownsMethod**: Since it is not possible to change how the covariances are approximated, **TFisher** should only be used when combining two-sided p values.

Finally, a rather different approach to combining p values is implemented in the **harmonicmeanp** package (Wilson 2019). In essence, the method uses the harmonic mean of the

p values,[12]

$$\bar{p} = \frac{k}{\sum_{i=1}^{k} 1/p_i},$$

based on which an asymptotically exact combined p value is computed using the Landau distribution (for details, see Wilson 2019). The method is robust to positive dependence among the p values and does not require any further input other than the p values to be combined:

```
R> harmonicmeanp::p.hmp(pvals, L = k)

     p.hmp
0.01302627
```

Although significant, this result is more conservative than the one we obtained with Brown's method. Further work is necessary to compare the methods implemented in **poolr** with the approach implemented in the **harmonicmeanp** package.

6. Discussion

The **poolr** package provides a collection of methods for combining p values that may be correlated due to non-independence among the tests from which they were derived. The package contains functions for a variety of "base methods" for combining independent p values along with several adjustments to these methods to account for dependence. Furthermore, the package provides logical extensions of various adjustments that have been described only incompletely in the literature and hence offers a more complete set of procedures.

For example, the base methods can be adjusted based on an estimate of the effective number of tests. However, only a subset of these adjustments have been described in the literature (e.g., Galwey 2009). In particular, the combination of this adjustment approach with the inverse chi-square method, the binomial method, and Stouffer's method is to our knowledge novel. Moreover, while the pseudo replication approach has been proposed in combination with the inverse chi-square method and the Bonferroni correction (Lin 2005; Liu *et al.* 2010), it can also be used in combination with any of the other methods. Finally, the use of the Satterthwaite approximation in combination with the inverse chi-square method is to our knowledge novel.

In this article, we illustrated the use of the **poolr** package with a dataset that was originally collected for a candidate-gene study. The same methods would be equally applicable to GWAS and other types of genetic studies (e.g., microarrays, DNA methylation analyses), studies using imaging techniques (e.g., functional magnetic resonance imaging), and essentially any situation where interest is not focused on the significance of individual tests (e.g., for SNPs, voxel), but of larger functional or structural units (e.g., genes, pathways, brain regions).

In applications of these types, researchers often want to compute a combined p value for many such units. The total computation time can then become a major concern. Carrying out proper permutation tests for all units might in fact be computationally too demanding, at

[12] We assume equal weights here; for the more general equation allowing different weights to be applied to the p values, see Wilson (2019).

least for those without access to powerful hardware to parallelize the procedures. An obvious alternative approach – with a negligible computational burden – is to apply the Bonferroni method to the *p* values within each unit, but this is often overly conservative. The methods implemented in the **poolr** package can be more powerful and at the same time are much less computationally demanding than permutation tests. Even the approach that mimics the latter can be carried out in seconds (as opposed to minutes or hours) through the use of pseudo replicates.

Moreover, an advantage of the methods implemented in **poolr** is that they do not require access to the raw data. The methods can be applied as long as the *p* values are available and a matrix that reflects the correlations among the test statistics. How to obtain the latter however requires some thought. As we mentioned earlier, the correlations among the interchanging elements across analyses is often a good proxy. We can illustrate this point with a simple example. Suppose we test whether some continuous response variable of interest is associated with a number of dichotomous predictors. Let the correlation (i.e., the phi coefficient) between two of the predictors be:

```
R> p11 <- 0.40
R> p10 <- 0.20
R> p01 <- 0.10
R> p00 <- 0.30
R> (p11 * p00 - p10 * p01) /
+    sqrt((p11 + p10) * (p01 + p00) * (p11 + p01) * (p10 + p00))
```

```
[1] 0.4082483
```

Now we repeatedly simulate data for $n = 100$ individuals under this scenario and extract the test statistics. Their correlation is very close to that of the two predictors themselves:

```
R> n <- 100
R> Z <- replicate(10000, {
+    X <- rmultinom(n, 1, c(p11, p10, p01, p00))
+    x1 <- X[1, ] + X[2, ]
+    x2 <- X[1, ] + X[3, ]
+    y <- rnorm(n)
+    res1 <- RcppEigen::fastLmPure(cbind(1, x1), y)
+    res2 <- RcppEigen::fastLmPure(cbind(1, x2), y)
+    c(res1$coefficients[2] / res1$se[2],
+      res2$coefficients[2] / res2$se[2])})
R> cor(Z[1, ], Z[2, ])
```

```
[1] 0.4112019
```

A further assumption underlying several of the methods is that the test statistics follow a multivariate normal distribution. In the example above, we know that the test statistics actually follow t-distributions with df = 98. However, from a practical point of view, this is hardly distinguishable from a normal distribution. To examine whether the assumption of *joint* multivariate normality is acceptable here, Figure 4 provides a filled contour plot of the

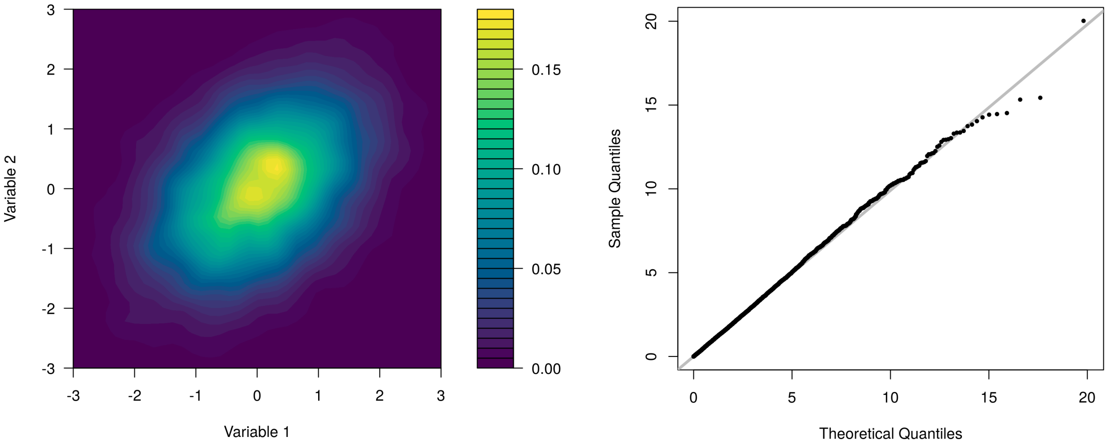

Figure 4: Filled contour plot showing the joint distribution of the test statistics and a QQ-plot of the Mahalanobis distances of the test statistics against the theoretical quantiles of a chi-square distribution with df = 2.

joint distribution of the two test statistics (based on two-dimensional kernel density estimation obtained via `MASS::kde2d()`) and a QQ-plot of the Mahalanobis distances of the test statistics against the theoretical quantiles of a chi-square distribution with df = 2 (which, under the assumption of joint multivariate normality, is the asymptotic distribution of the Mahalanobis distances; Healy 1968). Except for a few observations in the tail region, the assumption of multivariate normality appears to be an adequate approximation in this example.

Simulations for other types of scenarios and models (e.g., with continuous but skewed predictors, with interchanging response variables as opposed to interchanging predictors, or for logistic regression models) suggest that the findings above hold with some generality as long as the sample size is sufficiently large.[13] However, further work is required to delineate the specific circumstances under which (and how) the correlations among the interchanging elements can be mapped to the correlations among the test statistics.

For smaller sample sizes, the methods for converting the correlations among the test statistics into the covariances among the various target statistics (e.g., the $-2\ln()$-transformed p values needed for Brown's method) may need to be adjusted (e.g., Kost and McDermott 2002). In future versions, we plan on expanding the package with additional lookup tables for other cases besides the multivariate normal one.

We may also consider expanding the package with additional base methods for combining p values (and appropriate adjustments thereof to account for dependence). The choice of methods currently implemented was based on various considerations, including the attention the various methods have received in the literature. The methods by Fisher and Stouffer are often discussed, especially as alternatives to "proper" meta-analytic methods for combining evidence when only limited information about the studies/estimates to be combined is available (e.g., Laird and Mosteller 1990; Schmid, Koch, and LaVange 1991; National Research Council 1992; Piegorsch and Bailer 2009). Including the inverse chi-square method,

[13] For example, using logistic regression models with `x1` and `x2` as the response variables and `y` as the predictor in the simulation above yields essentially identical results.

these methods are also representative of the general class of methods that Loughin (2004) describes as "quantile combination methods", that is, they are all based on $\sum_{i=1}^{k} F^{-1}(p_i)$, where $F^{-1}(\cdot, 1)$ is the inverse of some cumulative distribution function (not necessarily for the chi-square distribution). In contrast, the Bonferroni and Tippett methods, and in a broad sense also the binomial test, belong to the class of "order-statistic combination methods" that make use of the fact that the ranked p values represent order statistics from a uniform distribution under the joint null hypothesis. The former two are particularly interesting due to their focus on the strongest "signal" (which may be of interest in certain circumstances) while the latter has the intuitive appeal of being a method that looks for "excess significance" in the set of p values.

Our choices were also influenced by methods that have received heightened attention in the genetics literature, given that the development of the **poolr** package was motivated, at least in part, by our work in this area. Aside from Fisher's, the Bonferroni, and Tippett's methods, this in particular led to the inclusion of the inverse chi-square method (e.g., see Liu *et al.* 2010, who discuss the use of this method in combination with the pseudo replication approach).[14] Although there are several other methods for combining p values (e.g., those proposed by Edgington 1972 and George 1977), they have not received as much attention as the aforementioned ones. Moreover, these two methods are currently only appropriate for combining independent p values and are already available in the **metap** package.

One issue we have not yet touched on is the possibility that R_x may not be a positive semi-definite matrix. Although we have assumed that the test statistics follow a multivariate normal distribution, in which case R_t must be non-negative definite by definition, we cannot observe the latter and instead use R_x to approximate R_t. Depending on how the elements in R_x are computed, this matrix may not be positive semi-definite. For example, suppose R_x contains the correlations among the interchanging elements across the analyses and some of the values of these variables are missing. If pairwise correlations are computed from different subsets of the data, then R_x is not guaranteed to be positive semi-definite. As a result, negative eigenvalues can arise, which has implications for the estimates of the effective number of tests described in Section 2.2. While the various estimators are defined in such a way that they can still be used in the presence of negative eigenvalues, it is currently unclear how negative eigenvalues should best be handled when using these adjustments. Therefore, whenever negative eigenvalues arise, the **poolr** package will issue a warning to make the user aware of this issue.

Moreover, such a correlation matrix would lead to an error when trying to generate multivariate normal random values for the empirically-derived null distributions. By default, **poolr** then uses a (slightly simplified) version of the `Matrix::nearPD()` function to find the nearest positive semi-definite matrix of R_x (Higham 2002) before proceeding with the computations (and issues a warning to inform the user that this algorithm was applied). In such a case, the same adjustment is also made when using the generalized methods (i.e., Brown's method, Strube's method, and the generalized inverse chi-square method). One can set argument `nearpd = FALSE`, in which case the use of this adjustment is disabled and the warning then turns into an error.

[14] Also, popular tests in this area (such as the Cochran-Armitage trend test, the allelic association test, or tests based on a dominant, recessive, or heterozygote (dis)advantage model) all lead to one degree of freedom chi-square tests (Ziegler and König 2010). Then (3) is simply the sum of these chi-square statistics, which therefore suggests itself as a natural way of combining the information across multiple SNPs.

Finally, we are fully aware of concerns about the overuse, overemphasis, or misinterpretation of p values and potential dangers that may arise from drawing dichotomous conclusions based thereupon (e.g., see Wasserstein and Lazar 2016, and the corresponding supplementary materials). Where possible, we therefore thoroughly support the call for increased emphasis on estimation in place of testing (with corresponding intervals that reflect the uncertainty of the estimates) and the use of meta-analytic methods to combine such estimates (Cumming 2012). However, in situations where we envision the use of the methods described in this paper (such as GWAS), there may not be any consistent directionality of the effects such that their combination using standard meta-analytic methods would be meaningful (e.g., the minor alleles at a locus may either be protective or harmful and this may differ across SNPs within a gene). Hence, methods based on the combination of p values may still have their place and we believe that the **poolr** package provides a useful collection of methods for this purpose.

References

Alves G, Hu YK (2014). "Accuracy Evaluation of the Unified p-Value from Combining Correlated p-Values." *PLOS One*, **9**(3), e91225. doi:10.1371/journal.pone.0091225.

Bates D, Eddelbuettel D (2013). "Fast and Elegant Numerical Linear Algebra Using the **RcppEigen** Package." *Journal of Statistical Software*, **1**(5), 1–24. doi:10.18637/jss.v052.i05.

Becker BJ (1994). "Combining Significance Levels." In H Cooper, LV Hedges (eds.), *The Handbook of Research Synthesis*, pp. 215–230. Russell Sage Foundation, New York.

Benjamini Y, Hochberg Y (1995). "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing." *Journal of the Royal Statistical Society B*, pp. 289–300. doi:10.1111/j.2517-6161.1995.tb02031.x.

Benjamini Y, Yekutieli D (2001). "The Control of the False Discovery Rate in Multiple Testing under Dependency." *The Annals of Statistics*, **29**(4), 1165–1188. doi:10.1214/aos/1013699998.

Billingsley P (1995). *Probability and Measure*. John Wiley & Sons, New York.

Birnbaum A (1954). "Combining Independent Tests of Significance." *Journal of the American Statistical Association*, **49**(267), 559–547. doi:10.2307/2281130.

Borchers HW (2021). ***pracma**: Practical Numerical Math Functions*. R package version 2.3.3, URL https://CRAN.R-project.org/package=pracma.

Brown MB (1975). "A Method for Combining Non-Independent, One-Sided Tests of Significance." *Biometrics*, **31**(4), 987–992. doi:10.2307/2529826.

Cheverud JM (2001). "A Simple Correction for Multiple Comparisons in Interval Mapping Genome Scans." *Heredity*, **88**(1), 52–58. doi:10.1046/j.1365-2540.2001.00901.x.

Cinar O, Viechtbauer W (2022). ***poolr**: Methods for Pooling p-Values from (Dependent) Tests.* R package version 1.1-1, URL `https://CRAN.R-project.org/package=poolr`.

Clopper CJ, Pearson ES (1934). "The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial." *Biometrika*, **26**(4), 404–413. `doi:10.1093/biomet/26.4.404`.

Conneely KN, Boehnke M (2007). "So Many Correlated Tests, So Little Time! Rapid Adjustment of p Values for Multiple Correlated Tests." *The American Journal of Human Genetics*, **81**(6), 1158–1168. `doi:10.1086/522036`.

Cousins RD (2008). "Annotated Bibliography of Some Papers on Combining Significances or p-Values." *Technical report*, University of California.

Cumming G (2012). *Understanding the New Statistics: Effect Sizes, Confidence Intervals, and Meta-Analysis.* Routledge, New York. `doi:10.4324/9780203807002`.

Dai H, Leeder JS, Cui Y (2014). "A Modified Generalized Fisher Method for Combining Probabilities from Dependent Tests." *Frontiers in Genetics*, **5**, 32. `doi:10.3389/fgene.2014.00032`.

DeGroot MH, Schervish MJ (2011). *Probability and Statistics.* 4th edition. Pearson, Boston.

Dewey M (2021). ***metap**: Meta-Analysis of Significance Values.* R package version 1.6, URL `https://CRAN.R-project.org/package=metap`.

Dudbridge F, Koeleman BP (2004). "Efficient Computation of Significance Levels for Multiple Associations in Large Studies of Correlated Data, Including Genomewide Association Studies." *American Journal of Human Genetics*, **75**(3), 424–435. `doi:10.1086/423738`.

Dunn OJ (1958). "Estimation of the Means of Dependent Variables." *The Annals of Mathematical Statistics*, **29**(4), 1095–1111. `doi:10.1214/aoms/1177706443`.

Edgington ES (1972). "An Additive Method for Combining Probability Values from Independent Experiments." *The Journal of Psychology*, **80**(2), 351–363. `doi:10.1080/00223980.1972.9924813`.

Ernst MD (2004). "Permutation Methods: A Basis for Exact Inference." *Statistical Science*, **19**(4), 676–685. `doi:10.1214/088342304000000396`.

Fisher RA (1932). *Statistical Methods for Research Workers.* 4th edition. Oliver and Boyd, Edinburgh. `doi:10.1007/978-1-4612-4380-9_6`.

Galwey NW (2009). "A New Measure of the Effective Number of Tests, A Practical Tool for Comparing Families of Non-Independent Significance Tests." *Genetic Epidemiology*, **33**(7), 559–568. `doi:10.1002/gepi.20408`.

Gao X, Starmer J, Martin ER (2008). "A Multiple Testing Correction Method for Genetic Association Studies Using Correlated Single Nucleotide Polymorphisms." *Genetic Epidemiology*, **32**(4), 361–369. `doi:10.1002/gepi.20310`.

George EO (1977). *Combining Independent One-Sided and Two-Sided Statistical Tests: Some Theory and Applications.* PhD dissertation, University of Rochester.

Goeman J, Solari A (2014). "Multiple Hypothesis Testing in Genomics." *Statistics in Medicine*, **33**(11), 1946–1978. doi:10.1002/sim.6082.

Good P (2013). *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypothesis*. Springer-Verlag, New York.

Härdle WK, Simar L (2015). *Applied Multivariate Statistical Analysis*. 4th edition. Springer-Verlag, New York. doi:10.1007/978-3-540-72244-1.

Healy MJR (1968). "Multivariate Normal Plotting." *Journal of the Royal Statistical Society C*, **17**(2), 157–161. doi:10.2307/2985678.

Higham NJ (2002). "Computing the Nearest Correlation Matrix: A Problem from Finance." *IMA Journal of Numerical Analysis*, **22**(3), 329–343. doi:10.1093/imanum/22.3.329.

Hochberg Y (1988). "A Sharper Bonferroni Procedure for Multiple Tests of Significance." *Biometrika*, **75**(4), 800–802. doi:10.1093/biomet/75.4.800.

Holm S (1979). "A Simple Sequentially Multiple Test Procedure." *Scandinavian Journal of Statistics*, **6**, 65–70.

Hommel G (1988). "A Stagewise Rejective Multiple Test Procedure Based on a Modified Bonferroni Test." *Biometrika*, **75**(2), 383–386. doi:10.1093/biomet/75.2.383.

Johnson RC, Nelson GW, Troyer JL, Lautenberger JA, Kessing BD, Winkler CA, O'Brien SJ (2010). "Accounting for Multiple Comparisons in a Genome-Wide Association Study (GWAS)." *BMC Genomics*, **11**(1), 724. doi:10.1186/1471-2164-11-724.

Kost JT, McDermott MP (2002). "Combining Dependent p-Values." *Statistics and Probability Letters*, **60**, 183–190. doi:10.1016/s0167-7152(02)00310-3.

Laird NM, Lange C (2011). *The Fundamentals of Modern Statistical Genetics*. Springer-Verlag, New York. doi:10.1007/978-1-4419-7338-2.

Laird NM, Mosteller F (1990). "Some Statistical Methods for Combining Experimental Results." *International Journal of Technology Assessment in Health Care*, **6**(1), 5–30. doi:10.1017/s0266462300008916.

Lancaster HO (1961). "The Combination of Probabilities: An Application of Orthonormal Functions." *Australian Journal of Statistics*, **3**(1), 20–33. doi:10.1111/j.1467-842x.1961.tb00058.x.

Lehne B, Lewis CM, Schlitt T (2011). "From SNPs to Genes: Disease Association at the Gene Level." *PLOS One*, **6**(6), e20133. doi:10.1371/journal.pone.0020133.

Li J, Ji L (2005). "Adjusting Multiple Testing in Multilocus Analyses Using the Eigenvalues of a Correlation Matrix." *Heredity*, **95**(3), 221–227. doi:10.1038/sj.hdy.6800717.

Lin DY (2005). "An Efficient Monte Carlo Approach to Assessing Statistical Significance in Genomic Studies." *Bioinformatics*, **21**(6), 781–787. doi:10.1093/bioinformatics/bti053.

Liu JZ, Mcrae AF, Nyholt DR, Medland SE, Wray NR, Brown KM, AMFS Investigators, Hayward NK, Montgomery GW, Visscher PM, Martin NG, Macgregor S (2010). "A Versatile Gene-Based Test for Genome-Wide Association Studies." *The American Journal of Human Genetics*, **87**(1), 139–145. doi:10.1016/j.ajhg.2010.06.009.

Loughin TM (2004). "A Systematic Comparison of Methods for Combining p-Values From Independent Tests." *Computational Statistics & Data Analysis*, **47**(3), 467–485. doi:10.1016/j.csda.2003.11.020.

Moskvina V, O'Dushlaine C, Purcell S, Craddock N, Holmans P, O'Donovan MC (2011). "Evaluation of an Approximation Method for Assessment of Overall Significance of Multiple-Dependent Tests in a Genomewide Association Study." *Genetic Epidemiology*, **35**(8), 861–866. doi:10.1002/gepi.20636.

Narasimhan B, Johnson SG, Hahn T, Bouvier A, Kiêu K (2021). ***cubature**: Adaptive Multivariate Integration over Hypercubes.* R package version 2.0.4.2, URL https://CRAN.R-project.org/package=cubature.

National Research Council (1992). *Combining Information: Statistical Issues and Opportunities.* National Academic Press, Washington, DC.

Nyholt DR (2004). "A Simple Correction for Multiple Testing for Single-Nucleotide Polymorphisms in Linkage Disequilibrium with Each Other." *The American Journal of Human Genetics*, **74**(4), 765–769. doi:10.1086/383251.

Phipson B, Smyth GK (2010). "Permutation p-Values Should Never Be Zero: Calculating Exact p-Values When Permutations Are Randomly Drawn." *Statistical Applications in Genetics and Molecular Biology*, **9**(1). doi:10.2202/1544-6115.1585.

Piegorsch WW, Bailer JA (2009). "Combining Information." *Wiley Interdisciplinary Reviews: Computational Statistics*, **1**(3), 354–360. doi:10.1002/9781118445112.stat03704.

Poole W (2021). ***EmpiricalBrownsMethod**: Uses Brown's Method to Combine p-Values from Dependent Tests.* R package version 1.22.0, URL https://www.bioconductor.org/packages/EmpiricalBrownsMethod/.

R Core Team (2021). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Resnick SI (2014). *A Probability Path.* Springer-Verlag, New York. doi:10.1007/978-0-8176-8409-9.

Rosenthal R (1978). "Combining Results of Independent Studies." *Psychological Bulletin*, **85**(1), 185–193. doi:10.1037/0033-2909.85.1.185.

Salyakina D, Seaman SR, Browning BL, Dudbridge F, Muller-Myhsok B (2005). "Evaluation of Nyholt's Procedure for Multiple Testing Correction." *Human Heredity*, **60**(1), 19–25. doi:10.1159/000087540.

Schmid JE, Koch GG, LaVange LM (1991). "An Overview of Statistical Issues and Methods of Meta-Analysis." *Journal of Biopharmaceutical Statistics*, **1**(1), 103–120. doi:10.1080/10543409108835008.

Schroeder MS, Culhane AC, Quackenbush J, Haibe-Kains B (2011). "**survcomp**: An R/**Bioconductor** Package for Performance Assessment and Comparison of Survival Models." *Bioinformatics*, **27**(22), 3206–3208. doi:10.1093/bioinformatics/btr511.

Šidák Z (1967). "Rectangular Confidence Regions for the Means of Multivariate Normal Distributions." *Journal of the American Statistical Association*, **62**(318), 626–633. doi:10.1080/01621459.1967.10482935.

Simes JR (1986). "An Improved Bonferroni Procedure for Multiple Tests of Significance." *Biometrika*, pp. 751–754. doi:10.1093/biomet/73.3.751.

Slatkin M (2008). "Linkage Disequilibrium: Understanding the Evolutionary Past and Mapping the Medical Future." *Nature Reviews Genetics*, **9**(6), 477–485. doi:10.1038/nrg2361.

Stouffer SA, Suchman EA, DeVinney LC, Star SA, Williams Jr RM (1949). *The American Soldier: Adjustment During Army Life (Vol. 1)*. Princeton University Press, Princeton.

Strube MJ (1985). "Combining and Comparing Significance Levels from Nonindependent Hypothesis Tests." *Psychological Bulletin*, **97**(2), 334–341. doi:10.1037/0033-2909.97.2.334.

Tippett LHC (1931). *Methods of Statistics*. Williams Norgate, London.

Van Assche E, Moons T, Cinar O, Viechtbauer W, Oldehinkel AJ, Van Leeuwen K, Verschueren K, Colpin H, Lambrechts D, Van den Noortgate W, Goossens L, Claes S, Van Winkel R (2017). "Gene-Based Interaction Analysis Shows GABA Ergic Genes Interacting with Parenting in Adolescent Depressive Symptoms." *Journal of Child Psychology and Psychiatry*, **58**(12), 1301–1309. doi:10.1111/jcpp.12766.

Van der Vaart AW (1998). *Asymptotic Statistics*. Cambridge University Press, New York.

Warnes G (2021). ***genetics**: Population Genetics*. R package version 1.3.8.1.3, URL https://CRAN.R-project.org/package=genetics.

Wasserstein RL, Lazar NA (2016). "The ASA's Statement on p-Values: Context, Process, and Purpose." *The American Statistician*, **70**(2), 129–133. doi:10.1080/00031305.2016.1154108.

Westfall PH, Young SS (1993). *Resampling-Based Multiple Testing: Examples and Methods for p-Value Adjustment*. John Wiley & Sons, New York.

Wilkinson B (1951). "A Statistical Consideration in Psychological Research." *Psychological Bulletin*, **48**(2), 156–158. doi:10.1037/h0059111.

Wilson DJ (2019). "The Harmonic Mean p-Value for Combining Dependent Tests." *Proceedings of the National Academy of Sciences of the United States of America*, **116**(4), 1195–1200. doi:10.1101/171751.

Yang JJ, Li J, Williams LK, Buu A (2016). "An Efficient Genome-Wide Association Test for Multivariate Phenotypes Based on the Fisher Combination Function." *BMC Bioinformatics*, **17**, 19. doi:10.1186/s12859-015-0868-6.

Yi L, Pachter L (2018). ***aggregation**: p-Value Aggregation Methods.* R package version 1.0.1, URL https://CRAN.R-project.org/package=aggregation.

Zhang H, Tong T, Landers JE, Wu Z (2020). "TFisher: A Powerful Truncation and Weighting Procedure for Combining p-Values." *The Annals of Applied Statistics*, **14**(1), 178–201. doi:10.1214/19-aoas1302.

Zhao JH (2007). "**gap**: Genetic Analysis Package." *Journal of Statistical Software*, **23**(8), 1–18. doi:10.18637/jss.v023.i08.

Ziegler A, König IR (2010). *A Statistical Approach to Genetic Epidemiology.* 2nd edition. John Wiley & Sons, Weinheim.

Affiliation:

Ozan Cinar
Department of Psychiatry and Neuropsychology
School for Mental Health and Neuroscience
Faculty of Health, Medicine, and Life Sciences
Maastricht University, The Netherlands
E-mail: ozan.cinar@maastrichtuniversity.nl

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
January 2022, Volume 101, Issue 1 *Submitted:* 2020-02-21
doi:10.18637/jss.v101.i01 *Accepted:* 2021-06-02